NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

VIDEO-TEXT PROCESSING
BY USING MOTOROLA 68020 CPU
AND ITS ENVIRONMENT

by

M. Kadri Hekimoglu

March, 1991

Thesis Advisor: Chyan Yang

Unclassified

security classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification Unclassified			1b Restrictive Markings
2a Security Classification Authority			3 Distribution/Availability of Report
2b Declassification/Downgrading Schedule			Approved for public release; distribution is unlimited.
4 Performing Organization Report Number(s)			5 Monitoring Organization Report Number(s)
6a Name of Performing Organization Naval Postgraduate School	6b Office Symbol *(if applicable)* EC		7a Name of Monitoring Organization Naval Postgraduate School
6c Address *(city, state, and ZIP code)* Monterey, CA 93943-5000			7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000
8a Name of Funding Sponsoring Organization	8b Office Symbol *(if applicable)*		9 Procurement Instrument Identification Number
8c Address *(city, state, and ZIP code)*			10 Source of Funding Numbers
		Program Element No	Project No / Task No / Work Unit Accession No

11 Title *(include security classification)* VIDEO-TEXT PROCESSING BY USING MOTOROLA 68020 CPU AND ITS ENVIRONMENT

12 Personal Author(s) M. Kadri Hekimoglu

13a Type of Report	13b Time Covered	14 Date of Report *(year, month, day)*	15 Page Count
Master's Thesis	From To	March 1991	149

16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
Field	Group	Subgroup	CPU, CRTC, DUART, Memory, MMU, PAL, EPLD, Video-Text-Generator.

19 Abstract *(continue on reverse if necessary and identify by block number)*

The objective of this thesis is to design a small, stand-alone microcomputer using the MC68020 CPU and its environment. It is dedicated to one of the most common jobs of microcomputers : **"Video-Text Generation"**. Therefore, it is named **"VTG (Video-Text Generator)"**.

VTG consists of a CPU (MC68020), CRT Controller (MC6845), and DUART (MC68681). The **CRT Controller** processes and generates the NTSC standard video-synchronization and video-text signals. The **DUART** accomplishes asynchronous serial communication with the **Keyboard unit** and allows for the parallel communication via its parallel port, with peripherals.

The VTG has four 32 KB of **RAM** for main memory and one 16 KB of **RAM** for the CRT Controller **Refresh Memory**. The system software and the initialization routine is saved in 32 KB of **ROM**. Memory management and the generation of chip control signals are accomplished by two **PALs (Programmable Logic Arrays)** and two **EPLDs (Erasable Programmable Logic Device)**.

The CPU, PALs, EPLDs, and Memory chips in the VTG work at a speed of 8 MHz, while the CRT Controller works at 2 Mhz.

In addition, the system has a **"Video Synchronization and Multiplexing Unit"** which make it possible to synchronize the VTG's video-text signal with an external video signal. Accomplishing this, the system can place its text information into any NTSC standard video picture.

To perform these jobs, the system does not need another microcomputer or aid. It can work as a stand-alone system.

In this thesis, the whole VTG system has been designed and implemented. Each PAL and EPLD was tested with a Logic Analyzer. Proper simulations were performed and observed to work properly, as they were programmed.

20 Distribution/Availability of Abstract	21 Abstract Security Classification	
☒ unclassified/unlimited ☐ same as report ☐ DTIC users	Unclassified	
22a Name of Responsible Individual Chyan Yang	22b Telephone *(include Area code)* (408) 646-2266	22c Office Symbol EC/Ya

DD FORM 1473,84 MAR 83 APR edition may be used until exhausted

security classification of this page
Unclassified

Approved for public release; distribution is unlimited.

Video-Text Processing
by using Motorola 68020 CPU
and its environment

by

M. Kadri Hekimoglu
Captain, Turkish Air Force
B.S.E.E., GAZI UNIVERSITY Ankara / TURKEY, 1976

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March, 1991

Author: _____
M. Kadri Hekimoglu

Approved by: _____
Chyan Yang, Thesis Advisor

_____
Frederick W. Terman, Second Reader

_____
Michael A. Morgan, Chairman
Department of Electrical and Computer Engineering

ABSTRACT

The objective of this thesis is to design a small, stand-alone microcomputer using the MC68020 CPU and its environment. It is dedicated to one of the most common jobs of microcomputers : **"Video-Text Generation"**. Therefore, it is named **"VTG (Video-Text Generator)"**.

VTG consists of a CPU (MC68020), CRT Controller (MC6845), and DUART (MC68681). The **CRT Controller** processes and generates the NTSC standard video-synchronization and video-text signals. The **DUART** accomplishes asynchronous serial communication with the **Keyboard unit** and allows for the parallel communication via its parallel port, with peripherals.

The VTG has four 32 KB of **RAM** for main memory and one 16 KB of **RAM** for the CRT Controller **Refresh Memory**. The system software and the initialization routine is saved in 32 KB of **ROM**. Memory management and the generation of chip control signals are accomplished by two **PALs (Programmable Logic Arrays)** and two **EPLDs (Erasable Programmable Logic Device)**.

The CPU, PALs, EPLDs, and Memory chips in the VTG work at a speed of 8 MHz, while the CRT Controller works at 2 Mhz.

In addition, the system has a **"Video Synchronization and Multiplexing Unit"** which make it possible to synchronize the VTG's video-text signal with an external video signal. Accomplishing this, the system can place its text information into any NTSC standard video picture.

To perform these jobs, the system does not need another microcomputer or aid. It can work as a stand-alone system.

In this thesis, the whole VTG system has been designed and implemented. Each PAL and EPLD was tested with a Logic Analyzer. Proper simulations were performed and observed to work properly, as they were programmed.

THESIS DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government. The views and opinions of the author expressed herein do not necessarily state or reflect those of the United States Government and shall not be used for advertising or product endorsement purposes.

TABLE O. CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to express my grateful thanks to Professor Chyan Yang and Professor Frederick W. Terman for their valuable help toward the accomplishment of this thesis, a product of many sleepless nights, and to dedicate it to my dear and patient wife Nesrin, and my unique children, Selin and Cagri.

As an indication of my love, I named the two PALs and two EPLDs used in this thesis after my wife Nesrin, my daughter Selin, my son Cagri, and little Selin, JR., who stayed for a long time with us.

I also would like to forward my special thanks to the Naval Postgraduate School E.C.E. Department for giving me this research opportunity, supplying all necessary test equipments and other facilities, and my appreciation to the Motorola Company and the AMD Company for providing the CPU and PAL chips used in the implementation of this thesis.

I. INTRODUCTION

As a result of recent improvements in electronics technology, microcomputers and microcomputer controlled systems have taken over jobs previously performed by manual methods in several areas such as automobile assembly, household systems, children's toys, etc. With modern VLSI technology, the design of such complex systems is more easily accomplished than ever. Another example is in video technology. Production of professional quality home video programs is easily accomplished utilizing microcomputer controlled smart video cameras or digital editing devices currently available. In recent years, video technology and microcomputer technology have become highly interrelated.

Because of these great improvements, there is a growing need to educate people on microcomputer technology. To educate students in this area, we need to teach and demonstrate topics using less complex systems. To be effective these educational systems must include the main features of the more sophisticated systems. However, many educational computer boards have been designed. The majority operate with a smart terminal, a host computer, or a local host computer network. These systems aid students in understanding the principles of microcomputer systems and/or system design considerations. One of the important concepts missing in the educational process is specifically the "CRT Controller System". When the student uses a smart terminal and writes code for CRT Controllers, the process is invisible. The main purpose of this thesis is to deal with these limitations and to show the integration of a CRT Controller, Keyboard Interface, and other parts of a microcomputer. Using this system, a student can program the CRT Controller to generate a video text signal using any video standard,

(i.e., NTSC, PAL) without any hardware modification, as well as communicate through the serial or parallel ports of the DUART used in the system.

In addition, the student can obtain a complete understanding of the design of a microcomputer system and its software. Furthermore, students can visualize the principles of video signal generation. Instructors will be able to incorporate this methodology into class projects to allow further investigation of other features of microcomputers.

II. THE BACKGROUND OF VIDEO-TEXT-GENERATION AND MC68020 CPU

A. THE CONCEPT OF THE VIDEO-TEXT-GENERATION

In this thesis, the processes used to generate text characters, lines, or pages to be placed on a video signal will be explained. "Text-Processing" should not be confused with "Word-Processing". For instance, it is very common to get video text pages that inform us of recent news events from TV broadcasts. To generate those text pages, some specific processes are required. The system, used to send those pages is somewhat different than the system addressed within this thesis. The technique used in those systems places the text information between the "interlace-balancing signals" of NTSC, PAL or any other commonly used video broadcast signal. These signals never disturb the normal video operation and are never seen on TV receivers or cable-TV monitors unless they are equipped to decode them. In other words, these "hidden" text pages can only be seen by a decoding system. The generation technique of the "hidden" text pages is similar to those used in this thesis.

In summary, this thesis addresses the generation of the text characters, the video-signal processing used to manage these text signals, and the design of the microcomputer system using the MC68020 CPU for processing tasks.

B. FEATURES AND ARCHITECTURE OF MC68020 CPU

1. The Architecture

The MC68020 CPU is an advanced microprocessor developed by Motorola. Although there are more advanced CPUs made by Motorola, such as the MC68030 and MC68040, the MC68020 maintains its popularity and is still used by many microcomputer manufacturers. The MC68020 is upwardly compatible with previous processors.

For example, the instruction set used on previous Motorola CPUs, such as MC68000 and MC68010, is compatible with the MC68020 instruction set. Furthermore, there are more instructions implemented in the MC68020.

The MC68020 microprocessor has a 32 bit address and data port. At each bus cycle, the microprocessor can select the port size of the external device, or the external device can have the MC68020 select the port size before any operand transfer. This very versatile feature is called "Dynamic Bus Sizing" and eliminates the all port size limitations. This feature also is very useful if the MC68020 is operated together with any 8-bit peripherals, communication interfaces, or CRT Controllers.

The 32-Bit port of MC68020 can address directly up to 4 Gigabytes [Ref. 1: pp.(1-6)-(1-7)]. Another important feature of the MC68020 is the "Cache Memory", not previously available on some models. It has a 128-word on-chip instruction cache memory, as compared to the Cache Memory in the MC68010 (3 words). This advantage reduces the total execution time and external bus activity without any degradation. This feature also makes it possible to prefetch an instruction stream once, and store it in on-chip cache memory, thus saving memory access times until another instruction stream becomes necessary. The cache memory is not used on the VTG (Video-Text Generator) implemented in this thesis, because the Logic Analyzer used for design and hardware verification cannot analyze the cache memory operation of the MC68020. Instead, the cache operation was disabled by connecting the "CDIS" (Cache Disable) pin of the MC68020 to ground.

Although the MC68020 contains over 200,000 transistors, it draws approximately 1.5 watts when used with 16 Mhz of clock rate. When it is working with internal cache, a throughput of 8 MIPS is possible. [Ref. 2: p.128]

2. Processing States

The MC68020 has three processing states. These are :

- Normal
- Exception
- Halt

In the normal state, the processor performs the normal instruction execution by fetching instructions and operands, storing the results, and communicating with external devices such as UART, CRT Controller, or Coprocessor.

In the exception state, interrupts, tracing, trap instructions, and other types of exceptions must be considered. If any unusual condition occurs during normal instruction execution, the exception processing state is entered. This state can also be entered by any hard reset, bus error, or an interrupt. The exception processing state allows the processor to easily and rapidly handle any unusual conditions.

Whenever a system failure occurs, the processor enters the halt state. In this state, no processor activity can take place. The only way to restart the processor activity is to reset the processor externally. Despite execution of a stop instruction, the processor can be restarted by executing a trace or interrupt. In contrast, if the halt state is entered, there is no way to restart the processor except by hard-reset.

These three states also have two privilege states :

- Supervisor Privilege State.
- User Privilege State.

The privilege of the supervisor state is higher than the user state. Therefore, all processor instructions can be executed in this state. The MC68020 has a somewhat distinctive supervisor state activities as compared to the other members of 68000 family.

Setting the M bit in the status register enables the supervisor stack space to be separated for either user tasks or interrupt associated tasks. This separation allows for increased efficiency in the multitasking operating system. Upon execution of an interrupt, the M bit is cleared. This is the default condition after a reset. The processor can be switched from the supervisor privilege state to user privilege state by execution of an instruction, which modifies the status register.

The S bit of the status register defines the User Privilege state. If the processor is executing instructions in the user state, the S bit of the status register is clear. These conditions are illustrated in Figure 1 on page 7. Most instructions can be executed at both the User or Supervisor privilege state. Some instructions, however, have important effects on the system and must be executed only at the supervisor privilege state. An example of these instructions are STOP or RESET.

Bus cycles for an instruction executed at the user privilege level are classified as user references. The values of the function codes on FC0-FC2 specify the user address space. While the processor is at the user level, it references to the system stack pointer implicitly, or if it addresses register seven (A7) explicitly, it refers to the user stack pointer (USP). [Ref. 1: pp. (4-1)-(4-4)]

3. Address Spaces

An address space is specified by the processor for each bus cycle with the function code signals. There are five types of address spaces which are encoded by the function codes :

- User Program Space
- User Data Space
- Supervisor Program Space
- Supervisor Data Space

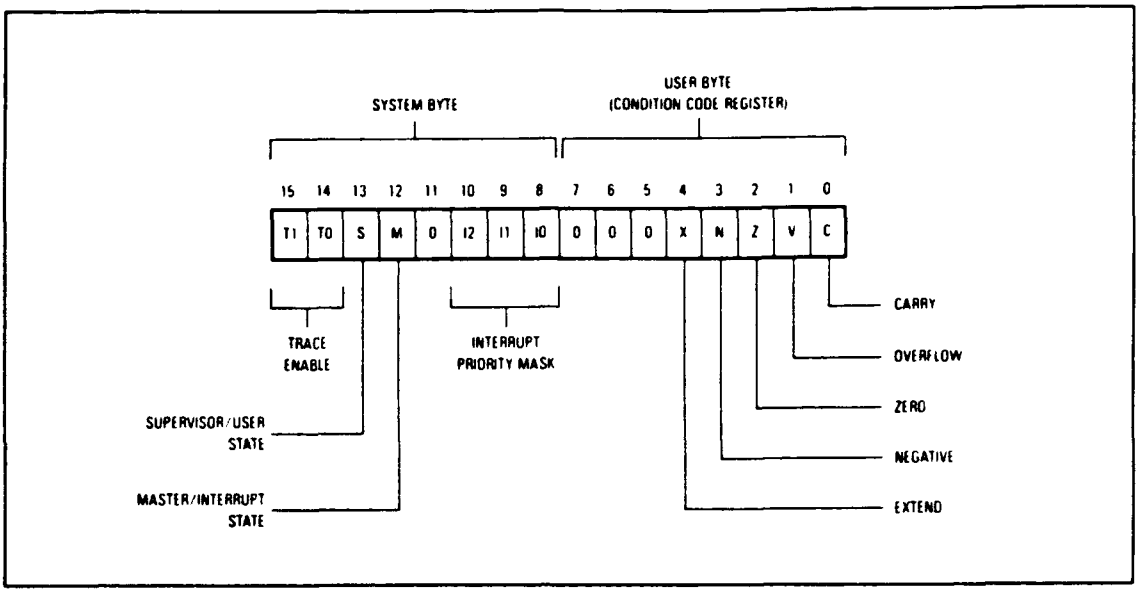


Figure 1. Status Register of the MC68020 (Copied from Reference 2)

- CPU Space

The other three address spaces are undefined. These address spaces are shown in Table 8 in the Appendix A. The memory locations of Supervisor and User address spaces are not predefined except for the addresses of the initial interrupt stack pointer and program counter values. During reset, the first two longwords beginning at memory location zero in the supervisor program space are used for processor initialization [Ref. 1: p. (4-5)].

The function code of S7 (B111) selects the CPU address space. This is an address space that does not include any instructions or operands except for the special CPU functions. The CPU space is only accessed for special purposes during communication between the processor and external devices. The 68000 uses this space for interrupt acknowledge cycles. The MC68020 also accesses the CPU for breakpoint acknowledge and coprocessor operations. The type of CPU space is specified by the

address lines A16, A17, A18, an. A19. On the Video-Text Generator, in this thesis, A18, A19, A20 address lines are used to decode the address spaces for CRT CONTROLLER, RAMs, ROM, Refresh Memory, and DUART.

4. Exception Processing

Exception processing on the MC68020 proceeds as follows :

- An internal copy of the status register is saved temporarily and the status register is set to process the exception. [Ref. 3: pp. 5-6]
- The exception vector is generated. An exception vector is a pointer to the memory location containing the address of the routine which handles the specified exception. There are 254 exception vectors available in the supervisor data space, and 2 vectors for the reset exception in supervisor program space. A group of 64 vectors is defined by the processor and the remaining 192 vectors are left for the user to define. Exception numbers can be generated externally or internally. [Ref. 3: pp. 5-6]
- The current processor context is saved on the exception stack frame created on the active supervisor stack. This context always includes the status register, the program counter, and the vector offset of the exception vector. Another field on the exception stack frame, called the "format field", is used to specify what additional processor state information has been put onto the stack frame, since there is more than one type of exception stack frame created by different exceptions. [Ref. 3: pp. 5-6]
- The address of the exception handler is then loaded into the program counter. The instruction at that address is fetched and the program execution is resumed. [Ref. 3: pp. 5-6]

III. DESIGN OF THE VIDEO TEXT GENERATOR

A. EXPLANATION OF THE BLOCKS OF THE VTG

The VTG (Video-Text-Generator) system consists of three major blocks :

- **CPU (Central Processing Unit) Block.** In this block CPU, Memories (two RAMs and one ROM), two PAL (Programmable Array Logic) and two EPLD (Erasable Programmable Logic Device that one of these was recently added to this block as a requirement), Reset timer circuitry, and DUART (Dual Asynchronous Receiver Transmitter) are placed.

- **CRTCON (CRT Controller Unit) Block.** In this block the CRT Controller, one 16 KB Memory Refresh RAM, one 16-Bit multiplexer unit consisting of four 4-Bit multiplexers, two 8-Bit edge triggered D flipflops, one 8-Bit transceiver (data direction selector), one Character-Generator ROM chip, one 4-Bit up/down counter (used as a dot counter), and one part of the EPLD_SELINJR (used as a combinational logic) are placed.

- **Video Synchronization and Multiplexing Block.** In this block one wide-band video amplifier, one voltage comparator, one PLL circuit, one DUAL Untriggerable One Shot multivibrator, and one 3-way analog switch chips are located.

Figure 2 through Figure 6 shows the schematics of these blocks. Figure 2 shows the previous CPU block. As seen in this schematic, a CD4017 decimal ring counter was used as a delay counter instead of the "EPLD_SELINJR". Deficiencies of CD4017 counter occurred, so this chip was replaced with the EPLD_SELINJR programmed for the same job. The latest schematic including this modification is depicted in Figure 3.

Figure 4 shows the sch matic of the previous design of the CRT Controller block. Because of the addition of the EPLD_SELINJR to the CPU Block, the combinational circuits on the previous design are also included in one part of the EPLD_SELINJR. So, two "EXOR" and three "AND" gates were used for text video output, and two "EXOR" and "one "AND" gates (used as "NOR" gate for the Text-Video Synchronization output) are included in the remaining pins of the EPLD_SELINJR. The latest schematic, including this modification, is shown in Figure 5.

The latest schematic of the Synchronization and Multiplexing block is shown in Figure 6.

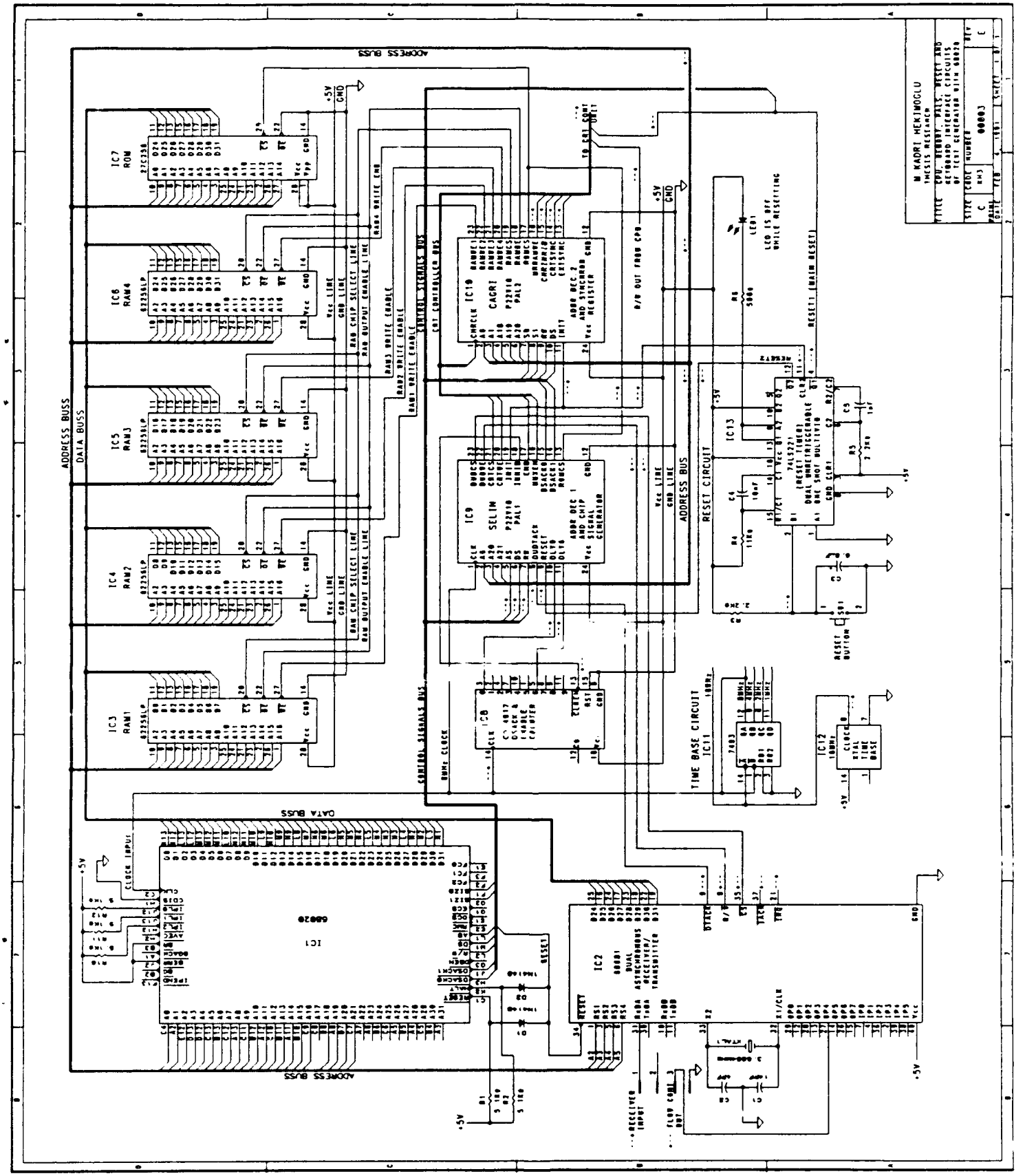


Figure 2. The schematic of the previous design of the CPU Block

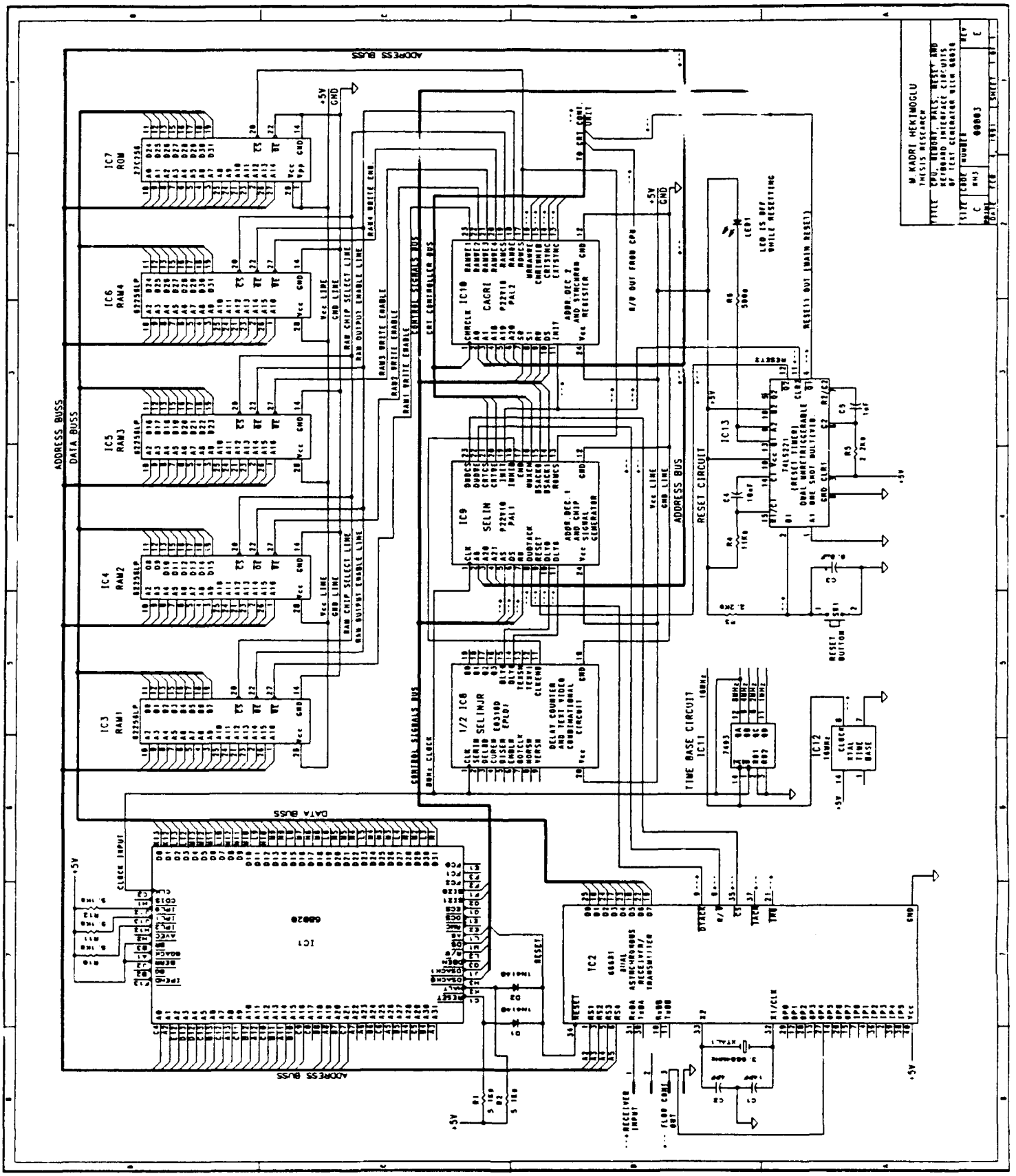


Figure 3. The schematic of the Modified CPU Block

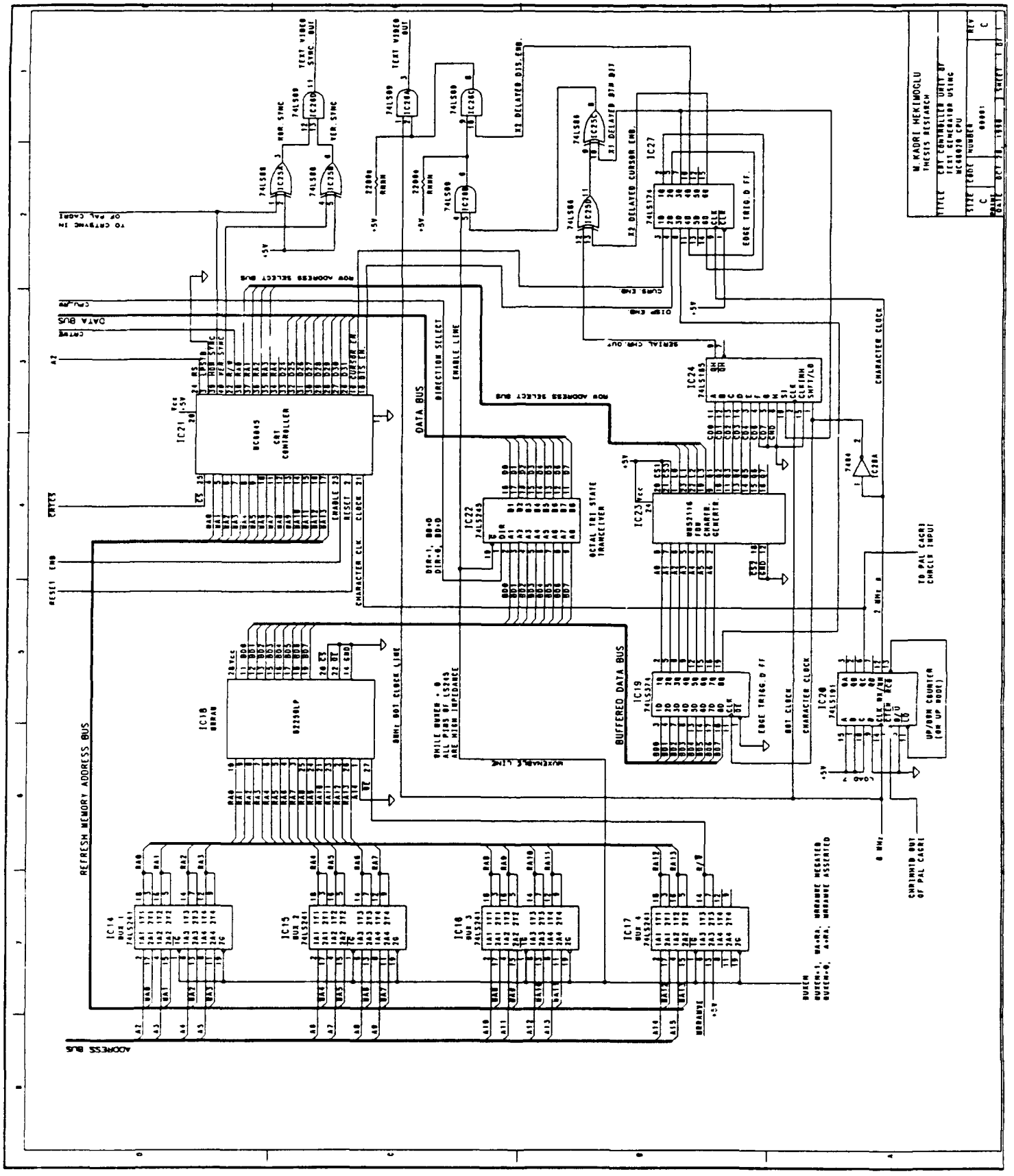


Figure 4. The schematic of the previous design of the CRT Controller Block

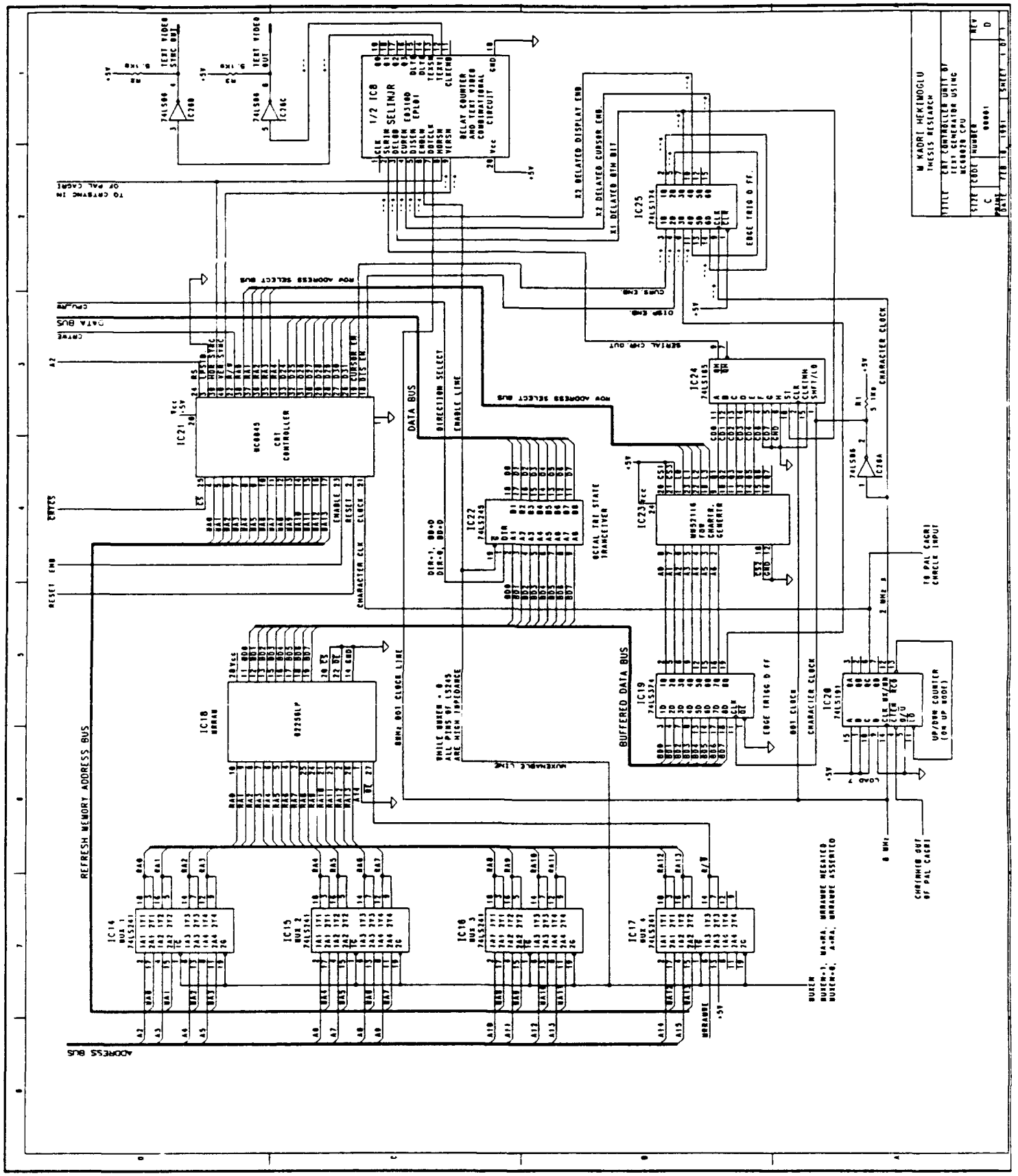


Figure 5. The schematic of the Modified CRT Controller Block

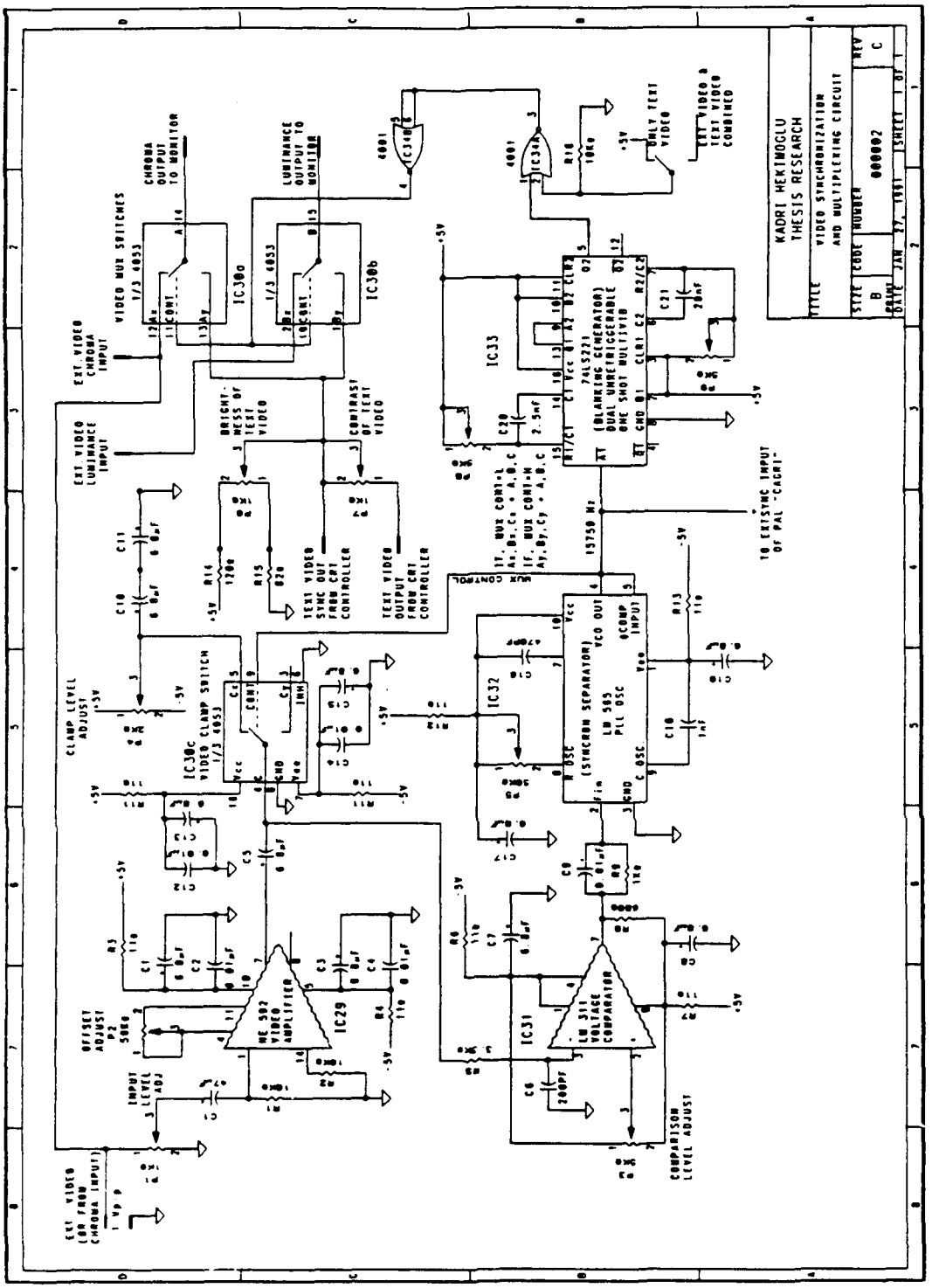


Figure 6. The schematic of the Synchronization and Multiplexing Block

B. DESIGN OF THE MEMORY MANAGEMENT AND CHIP CONTROL UNIT

Memory Management and the Chip Control Unit is an important part of the CPU Block. This unit generates the chip select and read/write signals for all the chips used in the system and processes some special chip control signals. The system operation will be directly affected if an incorrect design is implemented in this unit. Therefore, during the design phase of this project, each consideration was deeply and repeatedly checked to eliminate any future problem.

To program the PALs and EPLDs, the "Future Net-ABEL" software was utilized. The programs written for PAL_SELIN, PAL_CAGRI, EPLD_SELINJR, and EPLD_NESRIN are included in **Appendix D, E, F, G**. In these programs, as much as test vectors were used to check if the codes used to program those "Logic Devices" worked as expected. During the compilation phase of program, equations were three times minimized **(by "-r3" flag)** using the **"Presto"** algorithm. During the simulation phase, each code was verified using test vectors by ABEL software. After programming, all the PALs and EPLDs were tested under simulated conditions using **"Logic Analyzer"** and were verified as explained in Chapter IV.

1. Design Considerations of the MMU Chip Control Unit

a. PAL_CAGRI (Address Decoder-2 and Synchronous register)

The PAL_CAGRI generates all the RAM and ROM control signals. To write the codes for programming the PAL_CAGRI chip, the first step was to determine the memory spaces and draw a memory map (see Table 1 on page 17). According to this design consideration CAGRI.ABL program was written using the "ABEL" software as seen in Appendix E. In addition to "Address Decoding", PAL_CAGRI performs the synchronization job between text-video and exterior video, utilizing the **CHRINHIB** register. The implementation of the "CHRINHIB" register in PAL_CAGRI is shown Figure 7. A detailed explanation about the working of this register is provided in the

section "Verification of CHRINHIB (CHR Clock Inhibit Register) Output" of Chapter IV.

b. PAL_SELIN (Address Decoder-1 and Chip Signal Generator)

PAL_SELIN generates the CRT Controller and DUART Chip Selects and Write Enable signals. It was also used to generate the DSACK0 and DSACK1 bus completion signals until some miscalculations appeared. This job was then taken over by the programmed EPLD_NESRIN.

Additionally, PAL_SELIN generates the MUXEN (CRT Controller Multiplexer enable Signal), ENB (CRT Controller Enable Signal), INIT (Initialization Signal and INHIB (Delay Counter inhibit signal). The INHIB signal was no longer used after incorporating the EPLD_NESRIN.

To easily understand the design considerations of this unit we need to show the memory map of the system. The memory map of the system is illustrated in Table 1.

Table 1. THE MEMORY MAP OF THE VTG SYSTEM

	DURING INITIALIZATION	AFTER INITIALIZATION
RAM	00000 - 7FFFF	00000 - 7FFFF
ROM	00000 - 7FFFF	80000 - BFFFF
MRRAM	100000 - 13FFFF	100000 - 13FFFF
68681 DUART	200040 - 20007F	200040 - 20007F
6845 CRT CONTROLLER	200000 - 20003F	200000 - 20003F

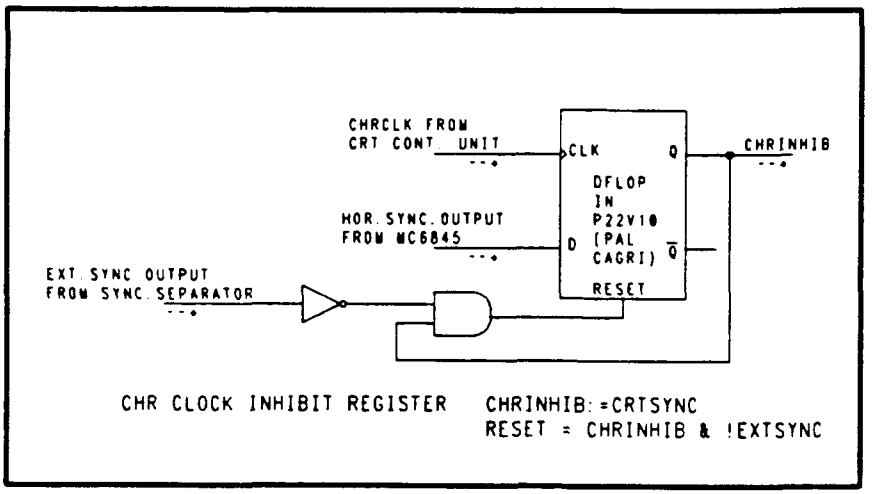


Figure 7. The implementation of the CHR Clock Inhibit Register in PAL_CAGRI

As seen in Table 1, the Memory Map of the ROM address space is changed after initialization. During initialization the RAM and ROM address spaces are the same. For copying the ROM to the RAM, this is necessary. When INIT signal is low, the ROM is located at the addresses between **00000 hex** and **7FFFF hex**. The CPU reads the initialization routine and the system software from the ROM at this address and writes the same address to the RAM. This is done by choosing the "high" R/W output of the CPU in the ROM Chip select equation **(see Appendix E)**. Therefore, the ROM Chip Select is asserted only during the READ cycle.

After initialization, the new ROM address space becomes **80000 hex - BFFFF hex**. This space is redundant for the address lines of the ROM. Since the ROM does not use the A15, A16, A17 address lines, the actual physical address space for the ROM during initialization is : **00000 hex - 7FFFF hex**; and after initialization, **80000 hex - 87FFF hex**.

The RAMs in the main memory do not use A17, and A18 address lines. Therefore, the actual address space for the RAMs during and after initialization is **00000 hex - 1FFFF hex.**

Since the MRRAM does not use A16 and A17 address lines, the actual address space for it is (during and after initialization) **100000 hex - 10FFFF hex.** We can easily understand the generating the chip select and write enable signals with Chip Signal Generating Table (see Table 2).

As can be seen from Table 2, the generation of RAM and ROM Chip select and Write Enable Signals are very straightforward. Two important signals at the generation of RAM Write Enable signals : are the A0 and A1 byte offset signals and the SIZ0 and SIZ1 bus cycle port size signals from the CPU. The usage of these signals are shown in Tables 3 and 4.

The RAMs work on the 32-Bit address space. In order to obtain the "dynamic bus sizing, it is necessary that the RAMWE signals be considered and handled using A0-A1 and SIZ0-SIZ1 signals. Therefore, RAM Chip controls can be designed to operate with any size of operands from 8-Bit to 32-Bit. The final design, incorporating these considerations, is in the **Appendix E, PAL_CAGRI.ABL** listing file.

Table 2. THE CHIP CONTROL SIGNAL GENERATION

	A21	A20	A19	A18	A6	INIT	R/W	A1	A0	S1	S0	DS
RAMCS	X	X	X	X	X	0	0	X	X	X	X	0
	X	0	0	X	X	1	X	X	X	X	X	0
RAMOE	X	0	0	X	X	1	1	X	X	X	X	X
RAMWE1	X	0	0	X	X	X	0	0	0	X	X	0
RAMWE2	X	0	0	X	X	X	0	0	1	X	X	0
	X	0	0	X	X	X	0	0	X	X	0	0
	X	0	0	X	X	X	0	0	X	1	X	0
RAMWE3	X	0	0	X	X	X	0	1	0	X	X	0
	X	0	0	X	X	X	0	0	1	X	0	0
	X	0	0	X	X	X	0	0	X	0	0	0
	X	0	0	X	X	X	0	0	X	1	1	0
RAMWE4	X	0	0	X	X	X	0	1	1	X	X	0
	X	0	0	X	X	X	0	1	X	1	X	0
	X	0	0	X	X	X	0	X	X	0	0	0
	X	0	0	X	X	X	0	X	1	1	1	0
ROMCS	X	0	0	X	X	0	1	X	X	X	X	0
	X	0	1	0	X	1	1	X	X	X	X	0
MRRAMWE	0	1	0	0	X	X	0	X	X	X	X	0
DUARTCS (DUDCS)	1	X	X	X	1	X	X	X	X	X	X	0
DUARTWE (DUDWE)	1	X	X	X	1	X	0	X	X	X	X	0
CRTCS	1	X	X	X	0	X	X	X	X	X	X	0
CRTWE	1	X	X	X	0	X	0	X	X	X	X	0

Table 3. THE SELECTION OF THE BUS CYCLE BYTE OFFSET

A1	A0	OFFSET
L	L	+0 BYTES
L	H	+1 BYTES
H	L	+2 BYTES
H	H	+3 BYTES

Table 4. THE SELECTION OF THE BUS CYCLE PORT SIZE

SIZE1	SIZE0	SIZE
L	H	BYTE
H	L	WORD
H	H	3 BYTES
L	L	LONGWORD

DSACK0 and **DSACK1** are also important chip control signals. These signals are inputs to the CPU to acknowledge the bus cycle completion and the size of the completed cycle. AS (Address Strobe) is the first sign of a bus cycle, and similarly, DSACK signals are the last signs of the bus cycle. If DSACK signals are not asserted after a bus cycle, that bus cycle can never be terminated. This is the most important

handshaking occurring in the MC68020. The usage of the DSACK signals is given in **Appendix A,** Table 7 on page 66.

The DSACK signal generation was changed after a miscalculation was discovered in the program of PAL_SELIN. This miscalculation and the programming of the EPLD_NESRIN are examined and explained in the section "Verification of DSACK Bus Cycle Completion Signals" of Chapter IV.

At this point The INHIB register is no longer used (as explained at the Chapter IV) in the section "Verification of the INHIB Output".

There are two more important implementations in PAL_SELIN : Enable Register and Initialization register. These are shown in the Figure 8, and Figure 9.

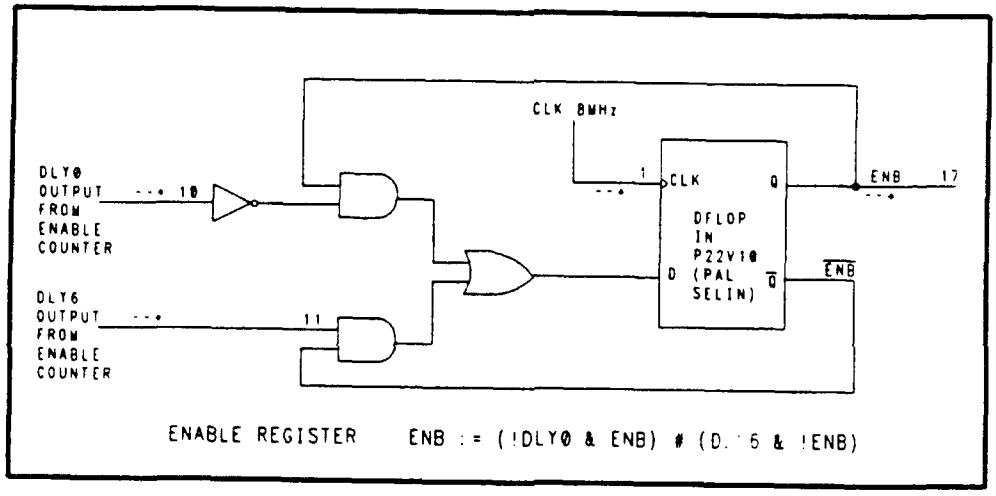


Figure 8. The implementation of the Enable Counter Register in PAL_SELIN

The operation of the Enable register has been explained in Chapter IV in the section "The verification of Enable Register".

c. Initialization of The VTG

The main idea about initialization has been taken from Tugcu [Ref.3: p. 92].

As seen in Figure 9, the Reset2 ($\overline{Q2}$) output of the reset timer unit is asserted and used to make the Init out low. After the first main reset, reset2 is triggered

and INIT output of the INIT Register goes to high. When the INIT output is low, the Initialization is started and continues until an access to DUART address space occurs. As soon as Duart Chip Select goes low, the INIT Register goes to high terminating the Initialization. Further information about this process can be found in Chapter IV, "Verification of INIT Register".

C. DESIGN AND IMPLEMENTATION OF DUART (KEYBOARD COMMUNICATION) UNIT

The design and implementation of DUART unit is straightforward. The implementation of the unit is shown in Figure 3. The unit has its own time base oscillator and can work asynchronously to produce its baud rates. The data lines D31-D24, belonging to the 8-Bit port of MC68020, are connected to the data line port of DUART unit. RS4-RS0 Register select address lines are connected to the MC68020's A2-A5

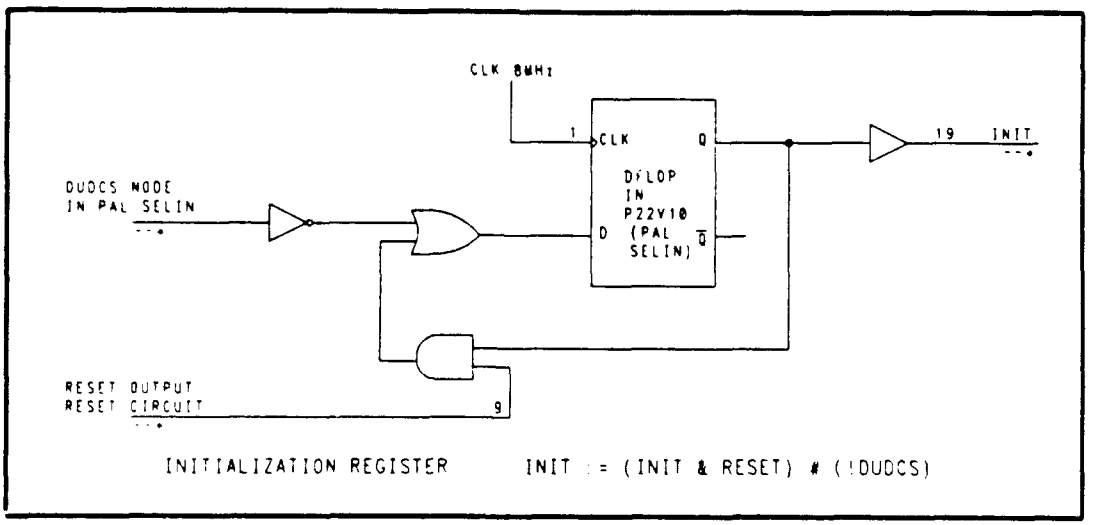


Figure 9. The implementation of the Initialization Register in PAL_SELIN

address lines. CS signals of this unit are generated in the PAL_SELIN unit and the reset line of DUART is connected to the main reset of the system.

The internal registers of this unit are set during the initialization according to the configuration requirements of baud rate or parity bits, etc. Two separate ports exist in order to communicate through the serial port of DUART. In the VTG port A was utilized to receive the serial data from the keyboard RxDA input, and to acknowledge the DUART buffer availability OP4 is used. Since no interrupt system is used on the VTG, the communication routine must check the output buffers of the DUART during each turn. The serial data received from the keyboard is read by the CPU as parallel data and is processed there. As considered in Chapter V, the parallel I/O ports of the DUART can be used for parallel communication with any peripheral device. Therefore to operate this system only software is now necessary. Any type configuration may be done by appropriate software since all necessary hardware connections and signals were supplied in the system.

D. DESIGN OF THE CRT-CONTROLLER SYSTEM

The schematics of the CRT Controller is given in Figure 4 and 5. No operational differences exist between these two schematics. However, in the latter schematic, the gates used at the output stages of CRT Controller were integrated in an EPLD, also used as a delay counter at the CPU block of the system as well. There is no relation between the part used in the CRT controller block and the part used in the CPU block. For simplicity the first schematic is followed for explanations of the gates.

For design and implementation of this block, we utilize the block diagrams and the explanations given by the Slater are used [Ref. 4: pp. 377-390]. The essential block diagram for the CRT Controller is shown in Figure 10.

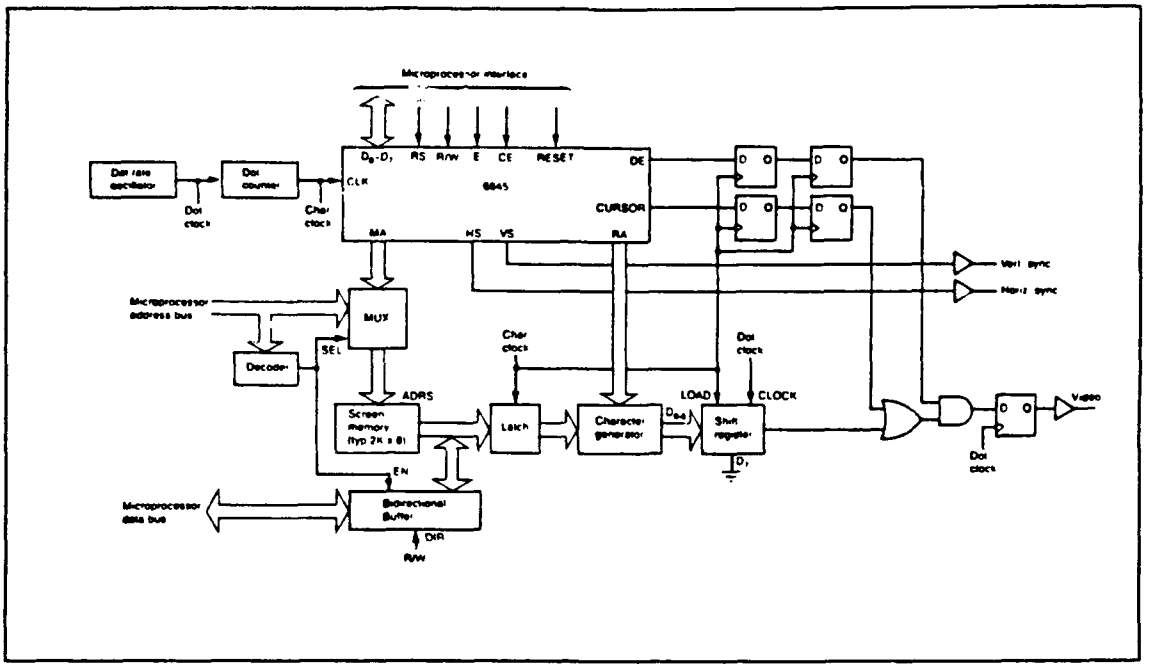


Figure 10. The essential block diagram of MC6845 CRT Controller (Copied from Reference 6)

As shown in Figure 10, there is a screen memory which is called **"Memory Refresh RAM"** in our system. Both the CRT Controller and the CPU can access this memory through the **MUX (Multiplexer) unit**. CPU accesses this memory within the time periods as programmed, with the character data loaded and sent to the character generator ROM.

Later these data are readdressed and accessed by CRT Controller and switched to the latch in the same sequence as they were loaded into the memory. These data are then switched to the character generator via latch to obtain a smooth data flowing. The character generator's row addresses are controlled by the row address select outputs of the CRT Controller. The CRT Controller gives the appropriate row numbers to the character generator to generate the dots belonging to the current scan line. This is explained in Figure 11.

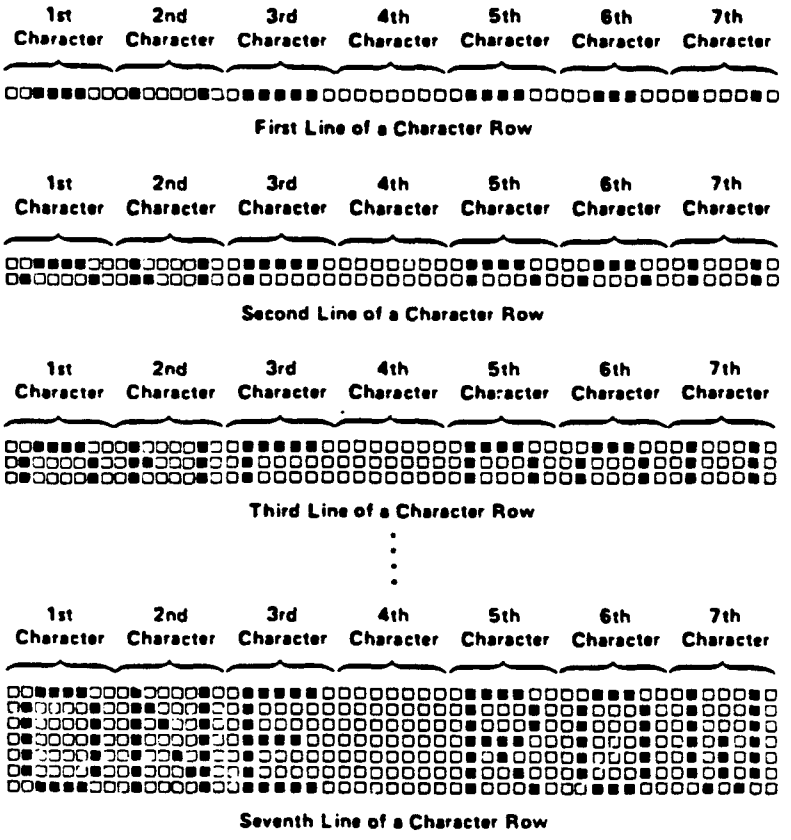


Figure 11. The generation of the text-video lines (Copied from Reference 4)

As seen in Figure 11, at the beginning of the each scan line, the number of the character row is entered to the Character Generator ROM by the CRT Controller. Meanwhile, the CRT Controller begins to increase the addresses from the left-most character on the screen. During this scanning period from the left to the right of the screen, at each character line period, the dots belonging to one row of that character sequentially appear on the screen. This line period equals the period between two horizontal synchronization pulses, and is 63.45 microseconds for the NTSC standard video signal. Repeating the same sequence for each character row, the whole text-line/lines

will be completed on the screen. The dot configuration of the character generator (NATIONAL SEMICONDUCTOR MM52116FDW) [Ref. 5: p.(9-17)] used in VTG is on the Figure 12. As we see from this figure, there are 9 rows and 5 columns for each line character lines. This means that the lines must be scanned 9 times to complete a video-text line.

The shift register forwards the character dots to the screen. The shift register is loaded by character clock rate (2 MHz) and shifted to the screen with the dot rate of 16 MHz in VTG.

Now, the explanation of the CRT Controller from the schematic of CRT Controller block in Figure 4 on page 13 can be continued. from the CPU and from the CRT Controller enter the **MUX** unit. The output of the **MUX** unit is selected by the **MUXEN** signal. MUXEN signal is low at the low part of the ENB (Enable) signal if MRRAM address space is selected. When MUXEN is low, the write enable of the MRRAM is asserted. Thus, the data from the CPU is stored in the MRRAM. When MUXEN is high, Memory Refresh addresses of the CRT Controller are enabled. At the same time, MUXEN signal cuts the connection between the data lines of CPU and MRRAM and since MUXEN is high, MRRAM begins to retrieve the data previously stored in it through the data lines to the IC19 edge triggered latch independently from the CPU. The data has then transferred to the character generator ROM at the character clock rate. At this time, the related row address information is sent to the character generator ROM. The character generator ROM retrieves the necessary dot data and sends them to the IC24 parallel to serial register. The dot information is loaded to the shift register with the character clock rate and shifted from the shift register at the dot rate to the IC25D EXOR. This video output is EXORed with the CURSOR ENABLE signal and 8th Bit of the IC19 latch's output. CURSOR ENABLE and DISPLAY ENABLE signals are delayed by IC27 since they are two clock cycles ahead of the text-video signal

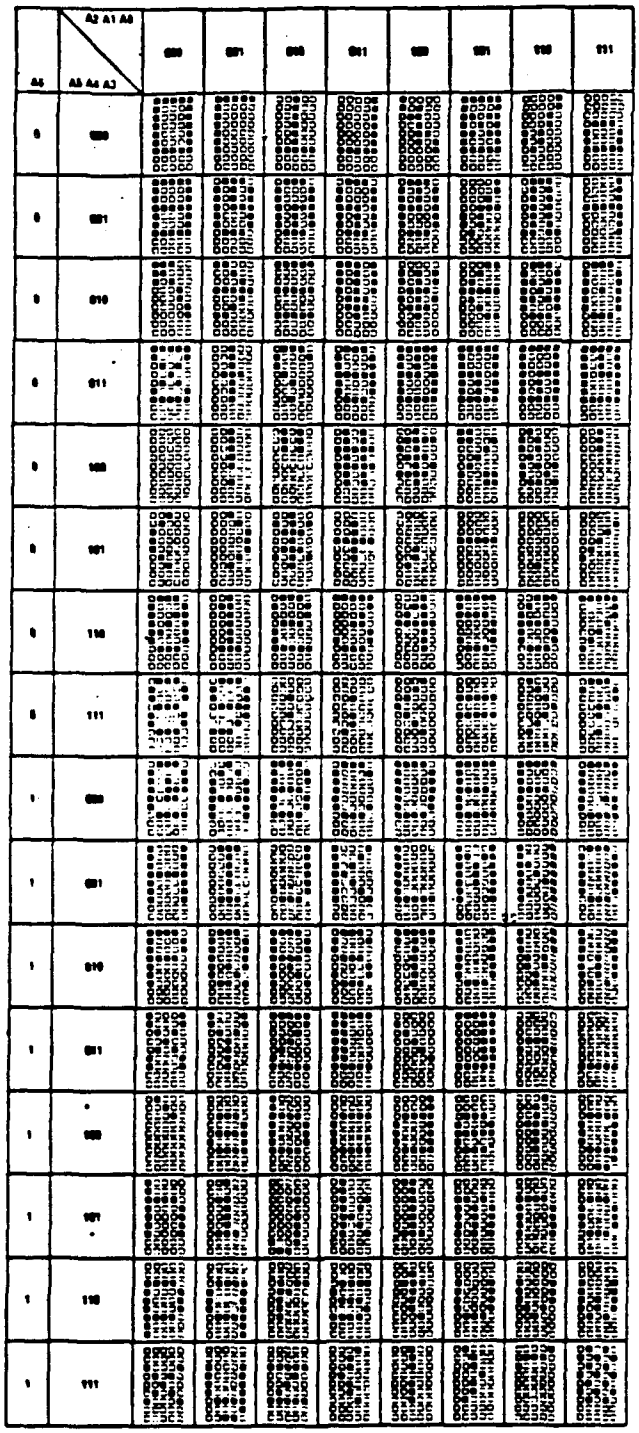


Figure 12. The dot configuration of the Character Generator on VTG (Copied from Reference 5)

from the shift register. Also the 8th bit of the IC19 is one clock cycle ahead of the video signal and is delayed for one clock cycle by IC27 latch. The 8th bit of MRRAM output is fed to the serial input of the shift register to draw the lines on the screen by appropriately programming this bit. Otherwise, the 8th bit of the MRRAM would be wasted. The synchronization signal outputs of the CRT controller are active-high and there they are NORed by using two inverter and one AND gate. The output of this and gate gives an active low composite synchronization signal.

1. Programming the Internal Registers of MC6845 CRT Controller

Table 5. MC6845 INTERNAL REGISTER ASSIGNMENT (COPIED FROM REFERENCE 6)

CS	RS	Address Register					Register #	Register File	Program Unit	Read	Write	Number of Bits							
		4	3	2	1	0						7	6	5	4	3	2	1	0
1	X	X	X	X	X	X	X	–	–	–	–								
0	0	X	X	X	X	X	X	Address Register	–	No	Yes								
0	1	0	0	0	0	0	R0	Horizontal Total	Char.	No	Yes								
0	1	0	0	0	0	1	R1	Horizontal Displayed	Char	No	Yes								
0	1	0	0	0	1	0	R2	H. Sync Position	Char.	No	Yes								
0	1	0	0	0	1	1	R3	H. Sync Width	Char.	No	Yes								
0	1	0	0	1	0	0	R4	Vertical Total	Char. Row	No	Yes								
0	1	0	0	1	0	1	R5	V. Total Adjust	Scan Line	No	Yes								
0	1	0	0	1	1	0	R6	Vertical Displayed	Char. Row	No	Yes								
0	1	0	0	1	1	1	R7	V. Sync Position	Char. Row	No	Yes								
0	1	0	1	0	0	0	R8	Interlace Mode	–	No	Yes								
0	1	0	1	0	0	1	R9	Max Scan Line Address	Scan Line	No	Yes								
0	1	0	1	0	1	0	R10	Cursor Start	Scan Line	No	Yes		B	P			(Note 1)		
0	1	0	1	0	1	1	R11	Cursor End	Scan Line	No	Yes								
0	1	0	1	1	0	0	R12	Start Address (H)	–	No	Yes								
0	1	0	1	1	0	1	R13	Start Address (L)	–	No	Yes								
0	1	0	1	1	1	0	R14	Cursor (H)	–	Yes	Yes								
0	1	0	1	1	1	1	R15	Cursor (L)	–	Yes	Yes								
0	1	1	0	0	0	0	R16	Light Pen (H)	–	Yes	No								
0	1	1	0	0	0	1	R17	Light Pen (L)	–	Yes	No								

Note (1). Bit 5 of the Cursor Start Raster Register is used for blink period control, and Bit 6 is used to select blink or non-blink.

As seen in Table 5, in the MC6845, R0 through R17, 18 internal registers configure the operation of CRT Controller. When CS is asserted and RS (Register Select) input is low, the address register of the CRTC is accessed through the first 5 bits of

MC6845 data port. To access an internal register, when CS and RS is low, the number of register to be accessed is input into the address register. Then the RS input is selected high while CS is low to access this addressed register. During RS high, any 8 bit number can be input into the selected register to configure the CRT Controller.

E. DESIGN OF THE VIDEO PROCESSING (VIDEO-SYNCHRONIZATION AND MULTIPLEXING) UNIT

This block was designed and implemented to separate the horizontal synchronization signal from the composite video signal. Utilizing a dual one-shot multivibrator, a video displaying signal is produced from this separated synchronization signal. This signal is used to multiplex the Video-Text signal and the Exterior-Video signal on the same screen. Separate video (as chroma and luminance) or composite video signals can be handled by this block. A schematic of this block is shown in Figure 6.

The first unit in the block is a NE592 wide-band video amplifier. This amplifier is used as a buffer and input level adjuster. It also makes the offset adjustment of the input video possible. The output of this unit is fed to clamp circuitry, which clamps the zero reference of this video signal to any offset voltage value between -5 and + 5 volts. The clamp circuit consists of an analog switch and a clamp capacitor. It charges the clamp capacitor during the synchronization pulses and at other times when a charged voltage level is added to the level of the video signal. With the clamp circuit, the offset value of the output of NE592 always stays at the set value. This is because of the need to make voltage comparisons in the LM311 voltage comparator unit. This unit is adjusted by a P3 potentiometer so that the LM311 is triggered at the each synchronization pulse encountered. By C9 R9 low-pass filter, the high frequency components of the video signal are filtered before entering the PLL (Phase Locked Loop) unit. The PLL unit is tuned to the horizontal synchronization frequency used in NTSC system (15750 Hz).

The PLL unit does two jobs. When no video input occurs, the output of PLL unit has an oscillation which is used to keep the clamping the output signal of the video amplifier. Otherwise, when there is no output from LM311, there will be no clamp, and if there is no clamp, no voltage comparison will occur, causing the system will to go into a vicious cycle.

The PLL unit also gives a very sharp and stable output signal by using the output from voltage comparator. This is because the PLL unit is not affected by some intermittent peaks with frequencies different from the **"Locked Frequency"**. The "Locked Frequency" of the PLL is adjusted by P5 potentiometer.

The next unit is 74221 dual unretriggerable one-shot multivibrator. The first half of this unit is triggered by each horizontal pulse from PLL unit. Its duration is as long as necessary to cover the color-burst area of the video signal. That is, the falling this signal occurs at the beginning of the visible area of the video signal. The timing of this period is adjusted by P8. The second half of the 74221 is triggered upon the falling of the first signal from high to low. Once this signal is triggered, it stays at high until the end of the visible area on the video picture. This signal also falls to low almost 5 microseconds before the horizontal synchronization pulse.

By driving the output of this unit to the control inputs of the multiplexer analog switches, multiplexing of two video signals is enabled, one from exterior video supply, the other is the Text-Video. Since the text-video is synchronized with the horizontal synchronization pulses from the PLL in the PAL_CAGRI, the timing of multiplexing and text-video outputs match each other. If it is desired, by disabling the multiplexing (controlling the input of IC34A) only the Text-Video can be operated.

IV. HARDWARE VERIFICATIONS

All the PALs (programmable Logic Devices) and EPLDs (Erasable Programmable Logic Devices) have been verified to make sure if they were properly programmed according to the system requirements and if they are working satisfactorily. For this verification, a binary counter was used with a maximum bit number equal to the maximum input variables used in the PALs and EPLDs. This binary counter was connected to the input pins of the devices to be verified. The output of the logic devices has been connected to the Logic Analyzer HP1651A. The counter was operated at a clock rate of 16 MHz which is the highest available frequency on the Video-Text Generator (VTG), although the PALs and EPLDs can work at higher speeds of up to 30 MHz and higher.

The HP1651A logic Analyzer used at the verification was configured and set according to the requirements of the programmed device. As the device was tested, the Analyzer, if the device works properly as programmed, catches the set point and shows it in the middle of the screen. The Analyzer takes samples over a large period of time as compared to the maximum frequency used in the device to be verified. The highest frequency possible is 28.5 Mhz. The Analyzer stores the events up to the set point and after the set point, but portrays the events at the set point to the middle of the screen. The outputs of the PALs or EPLDs can be checked to see if the design requirements are fulfilled. Also, the time durations between events are measured using the X and 0 cursors on time axes graphically, or numerically from the screen. This feature is so helpful that the rising and falling time delays of the gates can be seen instantly from the Analyzer screen.

In following verifications, some miscalculations were figured out during the experimentation. Those were related with the DSACK0 and DSACK1 outputs from the

PAL_SELIN. One of them was that A21 was thought to be a part of the DSACK0 equation in the PAL_SELIN program. The reason of this miscalculation was to assume that the Memory Refresh RAM works longword wise (32-Bit). However, it works byte-wise distinguishing from the main memory. This address line was in both DSACK0 and DSACK1 equation. Therefore in the latest programmed EPLD (EPLD_NESRIN), DSACK0 and DSACK1 were reconsidered and programmed according to the bus completion requirements.

The other miscalculations were related with the bus completion consideration of the ROM and CRT controller. The ENB signal has to be free-running. This condition was obtained by connecting the "CLKINHIB" input of the EPLD_SELINJR to ground. But, using a free-running frequency, it was impossible to give six clock cycles of delay to the bus completion timing of the ROM using the EPLD_SELINJR delay counter. The reason is obvious : the ROMCS signal is asserted when the AS signal goes low (asserted), and since DLY0 and DLY6 signals are free-running, these signals and AS signal would inconsistently catch each other. Finally, the delay given to the bus completion of timing the ROM would be inconsistent and consequently be impossible to give six clock cycles of delay to the ROMCS signal beginning from the assertion of AS (Address Strobe). So, the implementation of a separate three bit binary counter in the new EPLD_NESRIN program was included. This counter has been reset continuously whenever the AS is high. Then when AS goes to low (while ROMCS is asserted), the counter begins to count from zero. When the QA, QB, QC outputs become high (binary seven) the DSACK0 will be asserted to complete the ROM bus cycle (see **Appendix G**).

The last mistake on the DSACK0 output was related to the CRT Controller-bus completion. The DSACK0 signal had to wait for the high to low edge of the ENB signal, for assertion. But it would never see that edge of the ENB signal. The correction of this problem is going to be considered next.

In Summary, recently programmed EPLD (EPLD_NESRIN); DSACK0, DSACK1, DLY1 and DLY2 signals were reconsidered and the necessary input signals ROMCS, MUXEN, DUDCS, and DUDTACK were included.

A. VERIFICATIONS OF THE PALS AND THE EPLD

1. VERIFICATION OF THE PAL_SELIN

a. Verification of the Enable Register

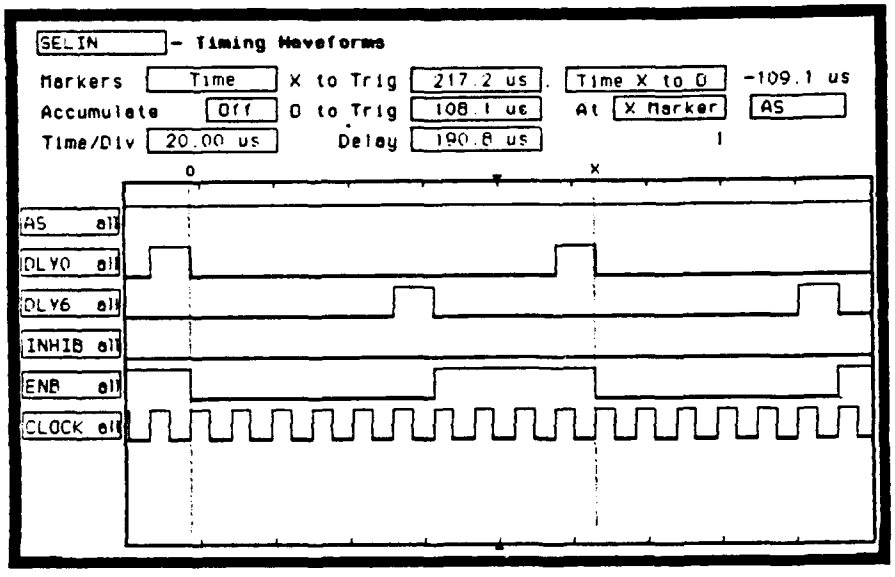


Figure 13. Verification of Enable counter in PAL_SELIN

As seen in Figure 13 when one cycle after DLY0 becomes high, the enable signal goes to low. After one clock cycle from the DLY6 became high, the enable signal goes to high. Therefore, the enable signal, which is the output of the enable register in PAL-SELIN, stays for six clock cycles at low, and four clock cycles at high. Previously it was designed to hold the counting of DLY counter in EPLD_SELINJR by detaining the CLKENB at high during AS (Address strobe) to low and DLY0 to high to make the synchronization possible between DLY signal and AS signal. However, this idea did not work because stopping the enable counter affects the data transfer directly from CPU to CRT CONTROLLER, causing it to disable the initialization phase. To recover this problem the CLK enable input of EPLD_SELINJR (pin 11) was grounded. Since the

synchronization is not so important between AS and ENABLE signals, DLY0 signal would not have to wait until AS goes to high.

b. Verification of the INHIB Output

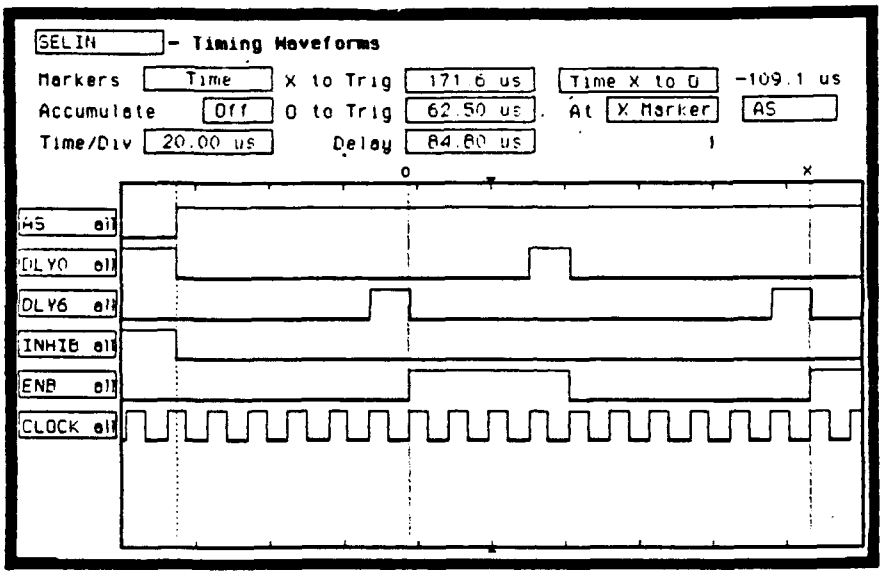


Figure 14. Verification of INHIB output in PAL_SELIN

As we explained before, this output is no longer used. However the verification is shown here. When AS is low and DLY0 output of EPLD-SELINJR is high, INHIB output becomes high. At this point DLY Counter would wait for AS = 1. During this phase the DLY counter would halt the counting. When AS becomes high, DLY counter counts up from 0 when INHIB goes to low. However, this was a logical error. It would be better if DLY counter counts whenever AS = 0. Therefore this implementation was then cancelled by grounding the CLK enable of the EPLD_SELINJR. We gave this up to synchronize the ENB signal with the AS signal as it was not a compulsory requirement. The printout of this verification is shown in Figure 14.

c. Verification of INIT (Initialization) Register

The implementation of the initialization circuit was shown in Chapter III, Figure 9. As mentioned before, there are two reset signals preceding the initialization signal. The first reset signal with a duration equal to 520 times the CPU clock rate is to reset the whole system. Since the clock frequency is 8 MHz, 520 times 125 ns equals to 65000 ns. This value in the system was adjusted as 70 microsecond. The second reset signal is asserted by going high to low upon the first one goes low to high. The second one drives the INIT register with a duration of 1.8 microsecond. In order to avoid from some consecutive INIT signals, the second reset signal is deleted when INIT signal is asserted. As seen in Figure 15, the duration of the second reset signal comes down to the 50 ns. This is just equal to the half clock cycle used in the simulation. When the half clock cycle after the reset2 is asserted, the INIT signal becomes consistently low and stays at low until the end of the initialization phase. When first read to the DUART by the CPU, this phase is terminated and INIT signal becomes high. This condition has been seen in the Figure 15. The assertion of the INIT signal is terminated upon AS = 0, A6 = 1, A21 = 1, by asserting the DUART chip select.

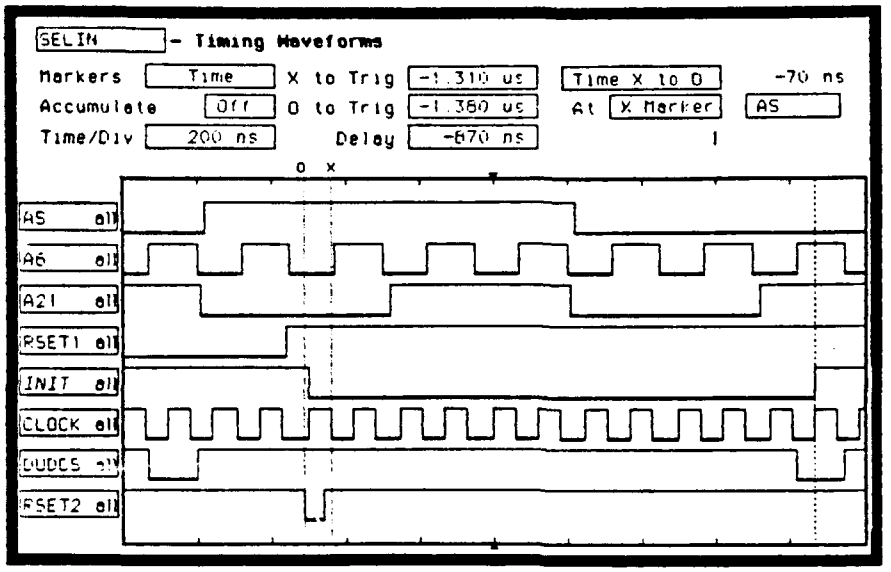


Figure 15. Verification of INIT register of PAL_SELIN

d. Verification of DSACK Bus Cycle Completion Signals

(1) The First Conditions of DSACK0 and DSACK1 outputs. The first conditions of the DSACK0 and DSACK1 are the same (see Appendix IV). When AS = 0, A20 = 0, ROMCS = 1, DUDCS = 1 CRTCS = 1, the both of the DSACK0 and DSACK1 outputs are asserted as seen in Figure 16. In this condition DSACK0 and DSACK1 will be asserted when no 8-Bit port is selected, that is, 32-Bit port is selected.

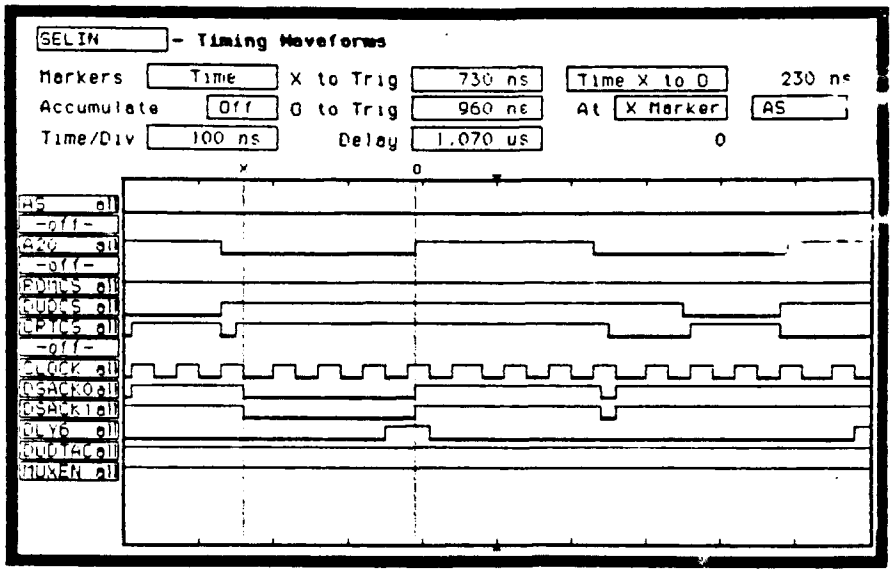


Figure 16. Verification of the First Conditions of DSACK0 and DSACK1 Bus Completion Signals

(2) The Second Condition of DSACK0 output. According to the second condition of the DSACK0 Bus Completion signal output when ROMCS=0 and DLY6=1, DSACK0 is asserted as seen in Figure 17. However, when ROMCS is asserted, AS goes to high. This is actually impossible because none of the memory select signals are asserted when AS=1. This is because the ROMCS signal has been generated in another PAL, "PAL_CAGRI". So, during simulation we drive the ROMCS input is driven to the PAL_SELIN directly from a bit output of the simulation counter and it leads to be seen as if there was a trivial result. The assertion of DSACK0 output in Figure 17 is shown between X and 0 cursors.

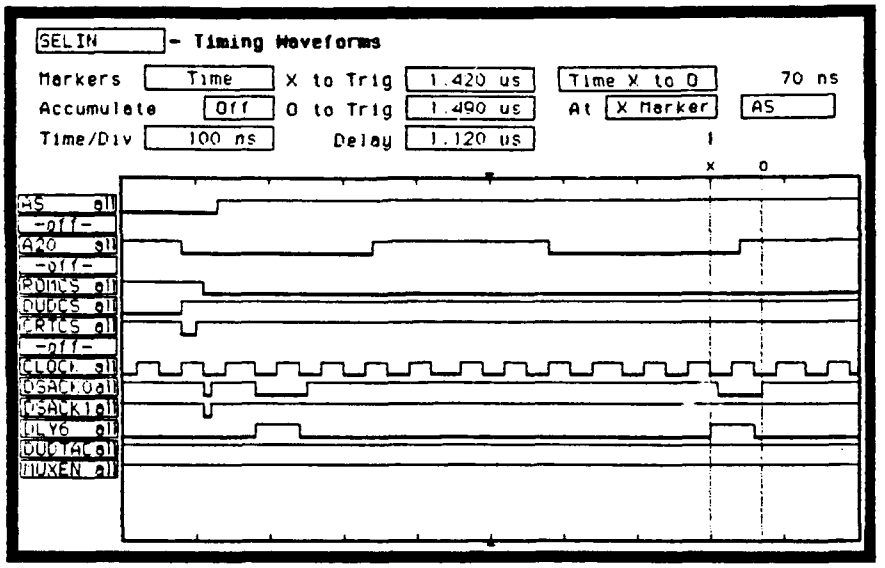


Figure 17. Verification of the Second Condition of DSACK0 Bus Completion Signal

(3) The Third Condition of DSACK0 Output. The third condition of the DSACK0 equation requires that if CRTCS is high and ENB signal is high, DSACK0 must be asserted. That means, if CRT Controller Chip is selected, during the high cycle of the ENB signal, the data transfer to or from the CRT Controller will be accomplished. This design consideration was also changed after a miscalculation was discovered. According to the data manual of MC6845 [Ref. 6: p. (1-166)], data transfer to or from the registers of MC6845 CRT CONTROLLER is performed at the high to low edge of the ENB (Enable) signal. If we would not change this condition, since we choose the high part of the ENB signal we would never have a chance to see the high to low edge of this signal. Because whenever ENB signal becomes high, DSACK0 signal would be immediately asserted attempting to terminate the bus cycle. In order to recover this problem, another EPLD (EPLD_NESRIN) has been programmed to select the DLY2 output for generating the DSACK0 signal by using the Q3-Q1 DLY counter outputs of the EPLD-SELINJR. Doing this allowed the high to low edge of the ENB signal to be

visible, as when the DLY2 output of the DLY counter is high, the ENB signal is still low. The DLY1 was not chosen because there may be some glitches during the high to low transition of the ENB signal. The ABEL program for this purpose is on the Appendix G. The printout of the previous implementation is shown in Figure 18.

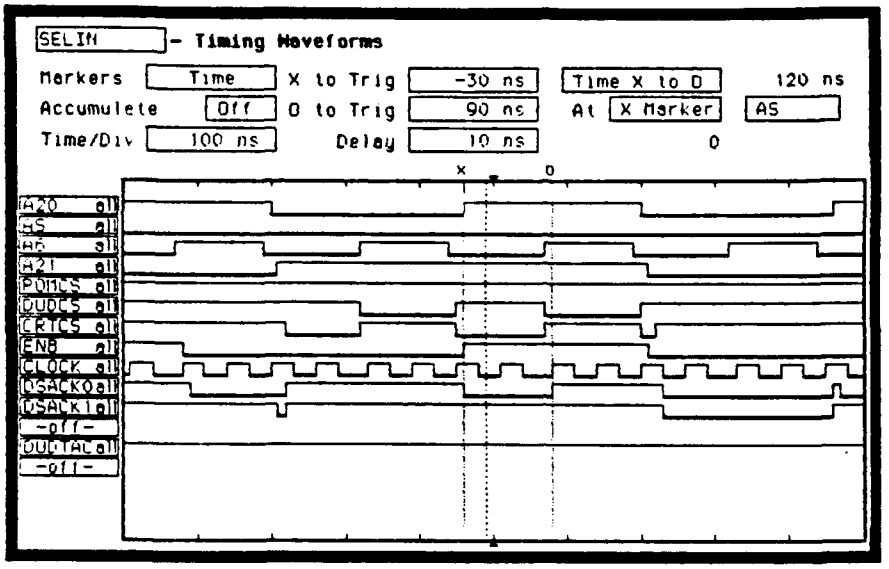


Figure 18. Verification of the third condition of the previous implementation of DSACK0 signal output

(4) The Fourth Condition of DSACK0 Output. This verification is related to the completion of the bus cycle of the DUART and straightforward. When DUDCS output is low (DUART chip select is asserted), DSACK0 signal waits for the DUDTACK (Duart Data Transfer Acknowledge) output to fall to low. When DUDTACK becomes low, DSACK0 signal terminates the DUART bus cycle by going to low. This is very straightforward because DUART has its own time base and the timing of the bus cycle is handled by itself. As seen in Figure 19, when A21 = 1, A6 = 1, AS = 0 (that means DUART chip select was asserted), when the DUDTACK signal goes to low, the DSACK0 signal becomes low by completing the bus cycle.

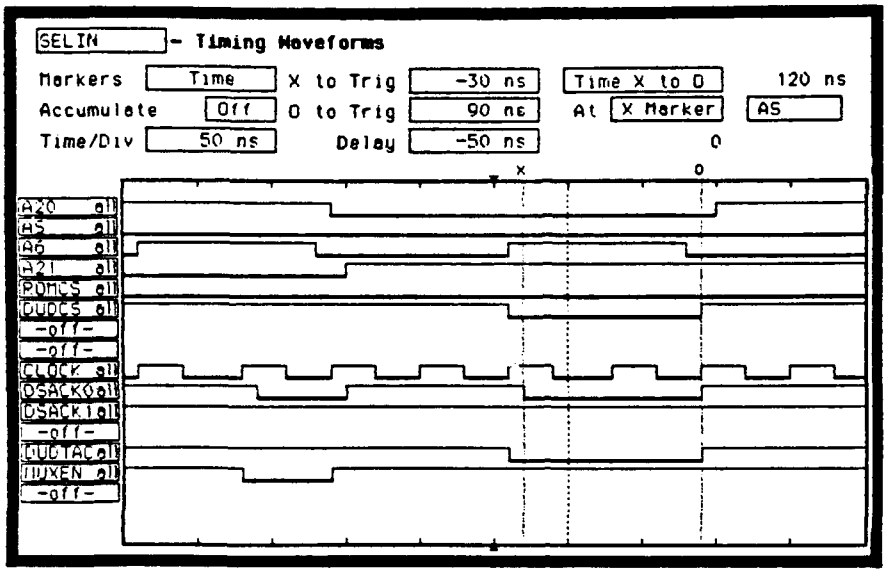


Figure 19. Verification of the fourth condition of the previous implementation of DSACK0 signal output

(5) The Fifth Condition of DSACK0 Output. This condition is to the complete the Memory Refresh Ram's bus cycle. According to the MUXEN equation at the PAL_SELIN program, when AS = 0, A20 = 1, A21 = 0, and ENB = 0, the MUXEN signal is asserted by going to low. The MUXEN signal enables the Write enable condition of the Memory Refresh RAM and directs the data flow from the CPU to MRRAM by asserting the G (direction control) input of the IC22 (Transceiver) in Figure 5. By including this signal in the DSACK0 equation, the signal can be terminated when MUXEN is low as no delay is necessary for the RAM with its access time as fast as (85 ns). The printout of this verification is show in Figure 20.

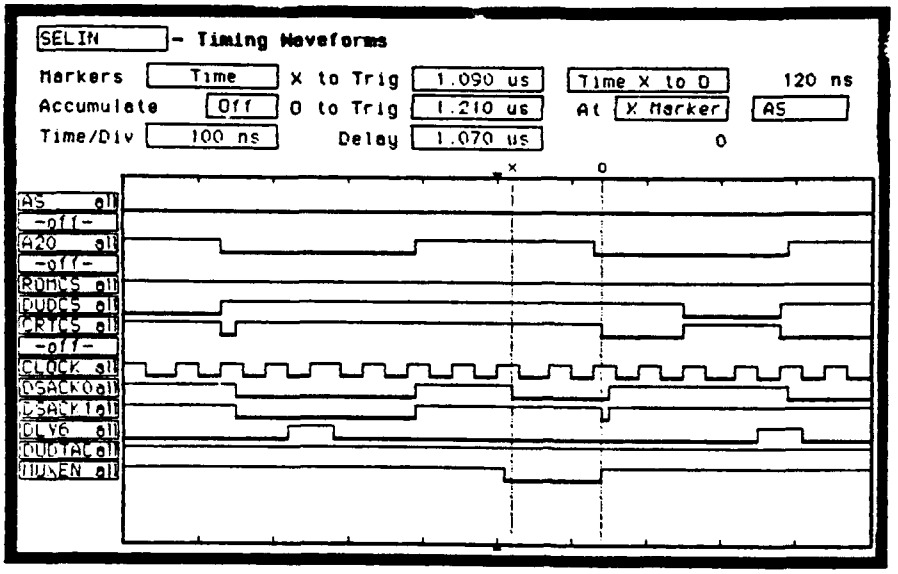


Figure 20. Verification of the fifth condition of the current implementation of DSACK0 signal output

e. Verification of MUXEN-CRT CONTROLLER Multiplexing signal Output

As we explained in the preceding section, when AS = 0, A20 = 1, A21 = 0 ENB = 0, the MUXEN (Multiplexer Enable) is asserted by going to low. This condition is verified as can be seen in Figure 21.

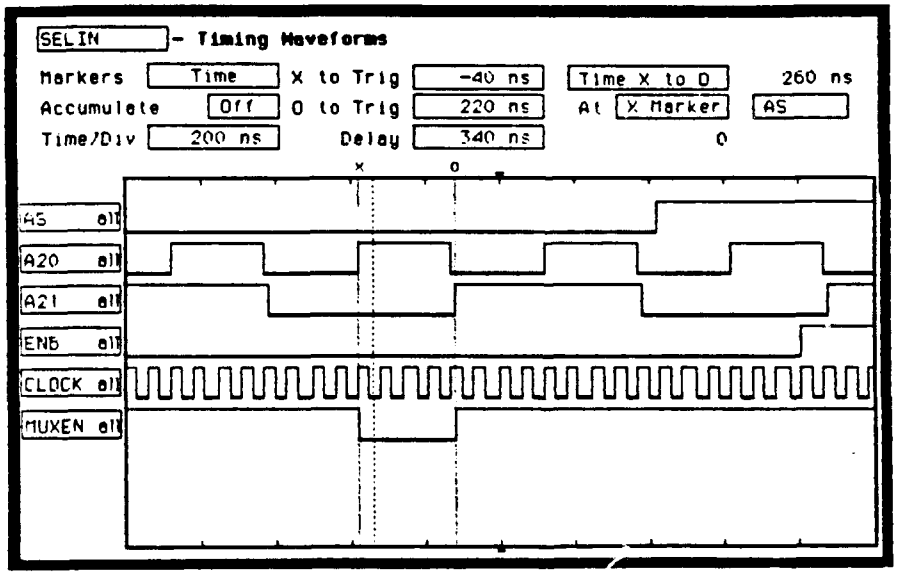


Figure 21. Verification of MUXEN (Multiplexer Enable signal) output

f. Verification of DUDCS (DUART Chip Select signal) Output

For assertion of the DUDCS, system requirements are A21 = 1, A6 = 1, and AS = 0. This is verified properly in Figure 19.

g. Verification of DUDWE (DUART Write Enable signal) Output

Verification also occurred that when A21 = 1, A6 = 1, DS = 0, RW = 0, as in the PAL_SELIN program, the DUDWE signal is asserted. This condition is shown in Figure 22.

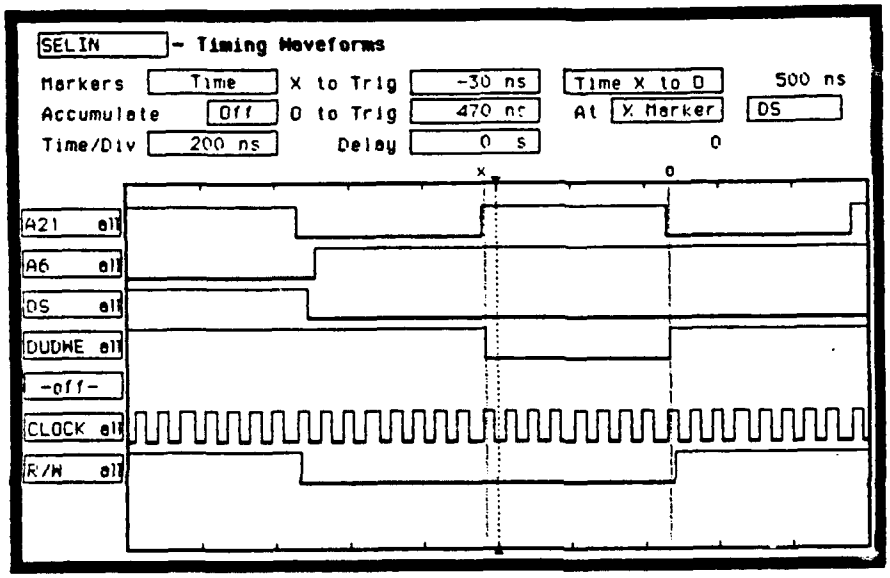


Figure 22. Verification of DUDWE (DUART Write Enable signal) Output

h. Verification of CRTCS (CRT CONTROLLER Chip Select signal) Output

In order to assert CRTCS, A21 = 1, A6 = 0, AS = 0, as according to PAL_SELIN program and design requirements. This verification is given in Figure 18.

i. Verification of CRTWE (CRT CONTROLLER Write Enable signal) Output

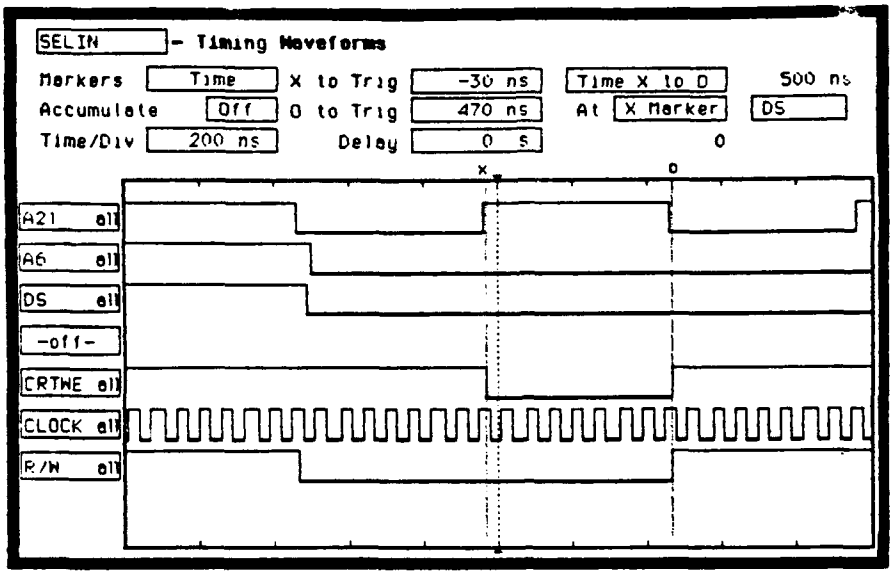


Figure 23. Verification of CRTWE (CRT Controller Write Enable signal) Output

As seen in Figure 23, A21 = 1, A6 = 0, DS = 0, RW = 0 to assert CTRWE. Therefore, the verification of this condition was obtained.

2. VERIFICATIONS OF THE PAL-CAGRI

In this section the printouts of verifications of the PAL_CAGRI are given. The details about the design and system requirements have been previously mentioned. Therefore, in this section, we are going to shortly explain the verifications. The requirements to be verified here, can be determined from the PAL-CAGRI program in Appendix E.

a. Verification of MRRAMWE (Memory Refresh RAM Write Enable signal) Output

For the verification of Write Enable of MRRAM (Memory Refresh RAM) it is necessary that A20 = 1, A19 = 0, A18 = 0, DS (Data Strobe) = 0, and RW = 0. It can be seen in Figure 24 that the verification meets this condition.

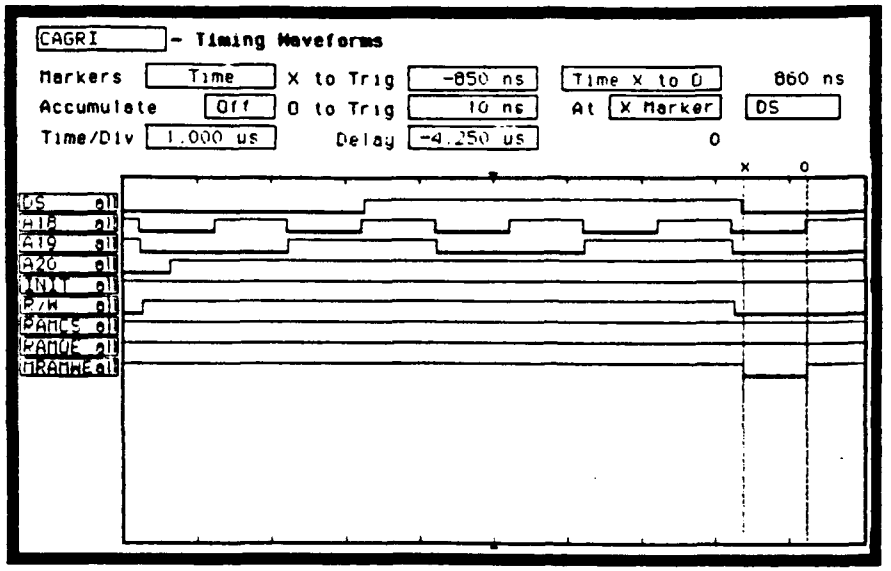


Figure 24. Verification of MRRAM (Memory Refresh RAM Write Enable Signal) Output

b. Verification of RAMCS (RAM Chip Select signal) Output

(1) The First Condition of RAMCS Output. In order to assert the RAMCS signal, the first condition must be A20 = 0, A19 = 0, INIT = 1, and DS = 0. As seen in Figure 25, the verification fulfills this condition.

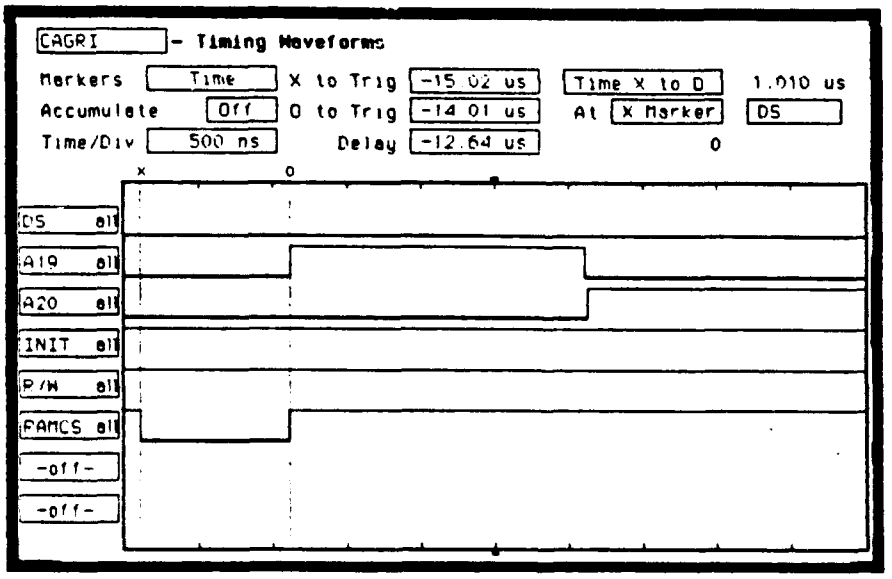


Figure 25. Verification of the first condition of RAMCS (RAM Chip Select) Output

(2) The Second Condition of RAMCS Output. This condition needs to be INIT = 0, RW = 0, and DS = 0. As seen in Figure 26, verification is satisfactory.

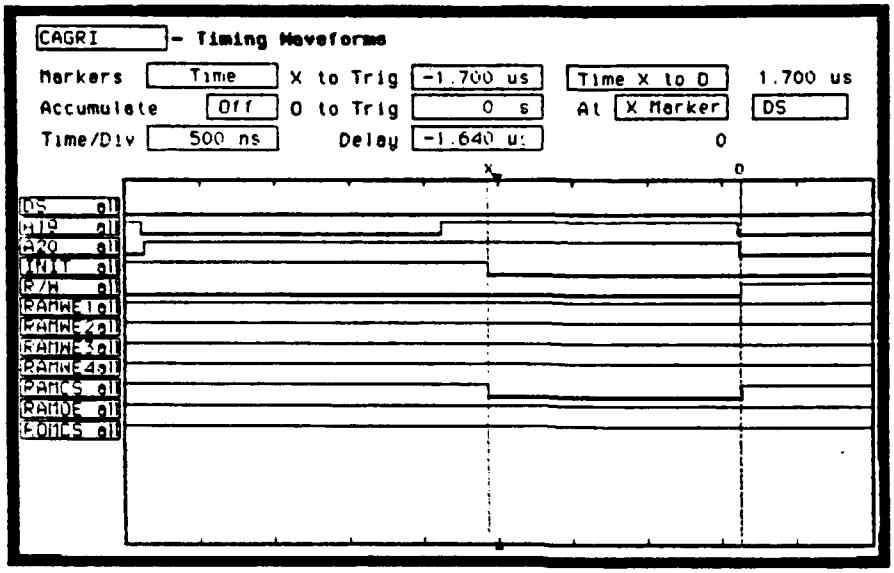


Figure 26. Verification of the second condition of RAMCS Output

c. Verification of RAMOE (RAM Output Enable Signal) Output

The verification of RAMOE (RAM output enable) needs to be A20 = 0, A19 = 0, INIT = 1, RW = 1, in order to assert this signal. Figure 27 shows us that the verification is satisfactory.

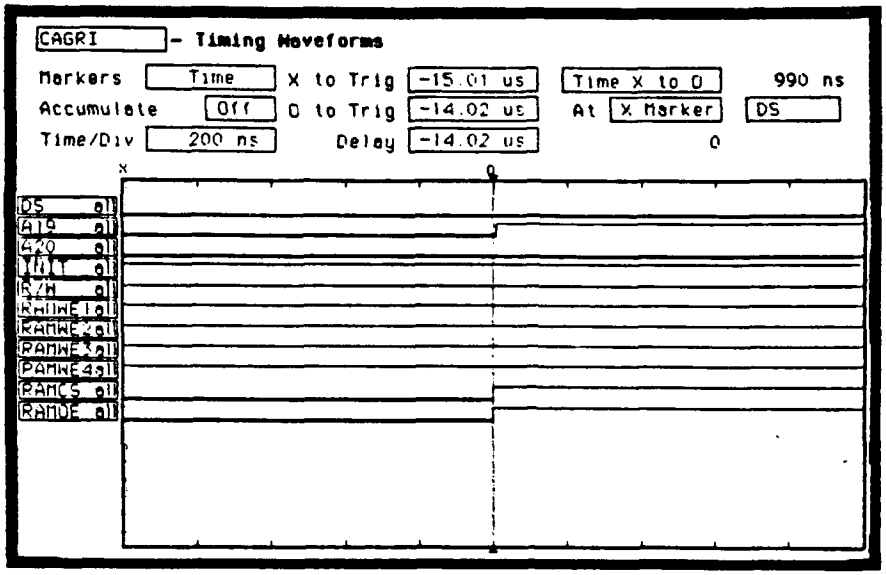


Figure 27. Verification of RAMOE (RAM Output Enable Signal) Output

d. Verification of RAMWE1 (RAM1 Write Enable Signal) Output

For the assertion of this signal, it must be A20 = 0, A19 = 0, A1 = 0, A0 = 0, DS = 0, and RW = 0. As seen in Figure 28, when this condition is met, the signal is asserted.

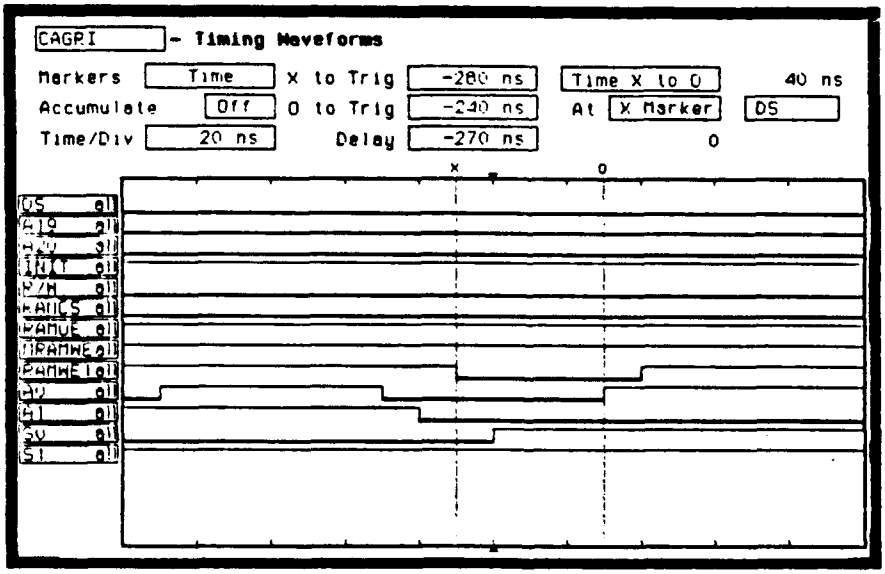


Figure 28. Verification of RAM1 Write Enable Signal Output

e. Verifications of RAMWE2 (RAM2 Write Enable Signal) Output

(1) The First Condition of RAMWE2 Output. When A20=0, A19=0, A1=0, S0(SIZ0)=0, DS=0, and RW=0 the RAMWE signal has to be asserted. The Figure 29 verifies that this condition has been met.

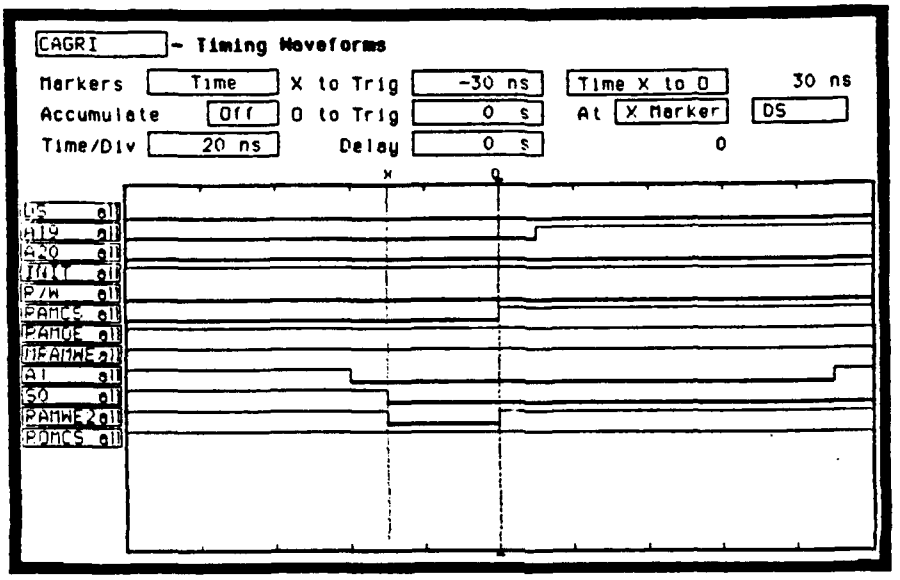


Figure 29. Verification of the first condition of the RAM2 Write Enable

(2) The Second Condition of RAMWE2 Output. The Figure 30 verifies that the RAM2 Write Signal Enable is asserted when A20=0, A19=0, A1=0, A0=1, DS=0, RW=0.

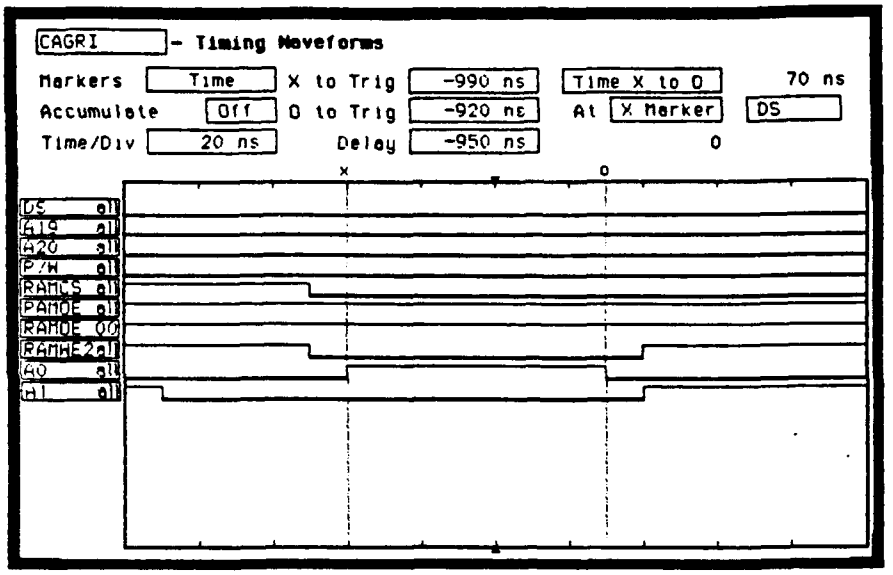


Figure 30. Verification of the second condition of the RAM2 Write Enable Signal

(3) The Third Condition of RAMWE2 Output. When A20 = 0, A19 = 0, A1 = 0, S1(SIZ1) = 1, DS = 0, and RW = 0, the RAM2 Write Enable is asserted. The verification of this condition is shown in the Figure 31.

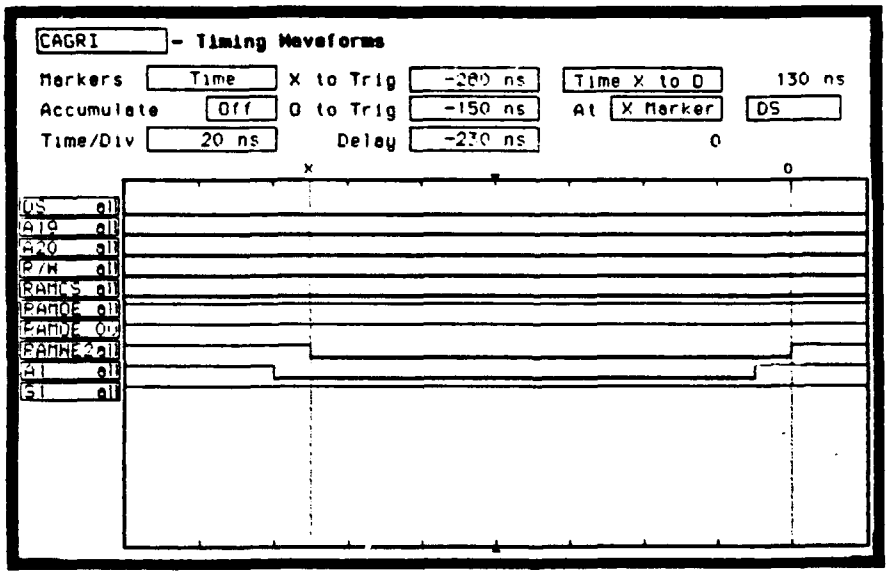


Figure 31. Verification of the third condition of the RAM2 Write Enable Signal

f. Verification of RAMWE3 (RAM3 Write Enable Signal) Output

(1) The First Condition of RAMWE3 Output. In order to meet the first condition of the RAMWE3, it must be A20 = 0, A19 = 0, A1 = 1, A0 = 0, DS = 0, RW = 0. As we see in Figure 32, this condition was verified.

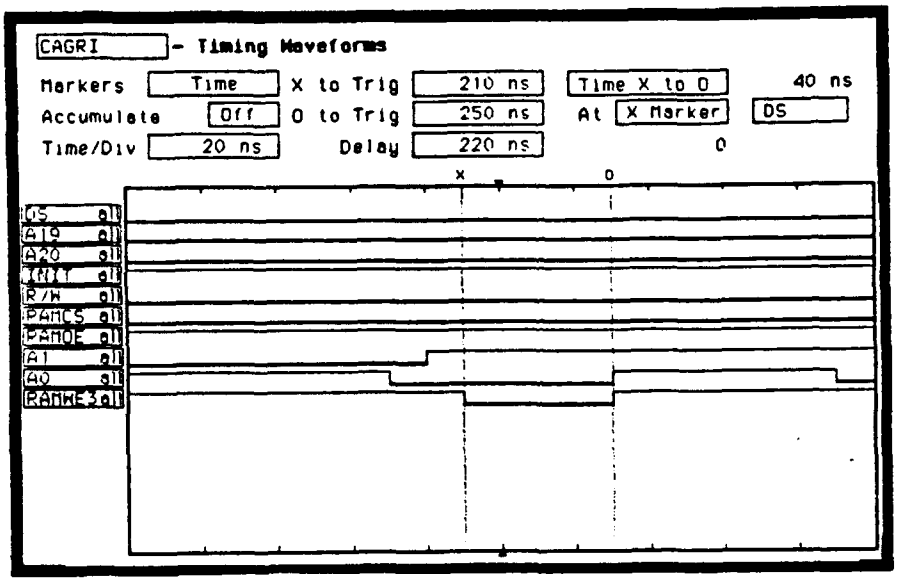


Figure 32. Verification of the first condition of the RAM3 Write Enable signal

(2) The Second Condition of RAMWE3 Output. The requirement in order to meet this condition is A20 = 0, A19 = 0, A1 = 0, S1(SIZ1) = 0, S0(SIZ0) = 0, DS = 0, and RW = 0. As seen in Figure 33, the assertion of this signal has been verified.

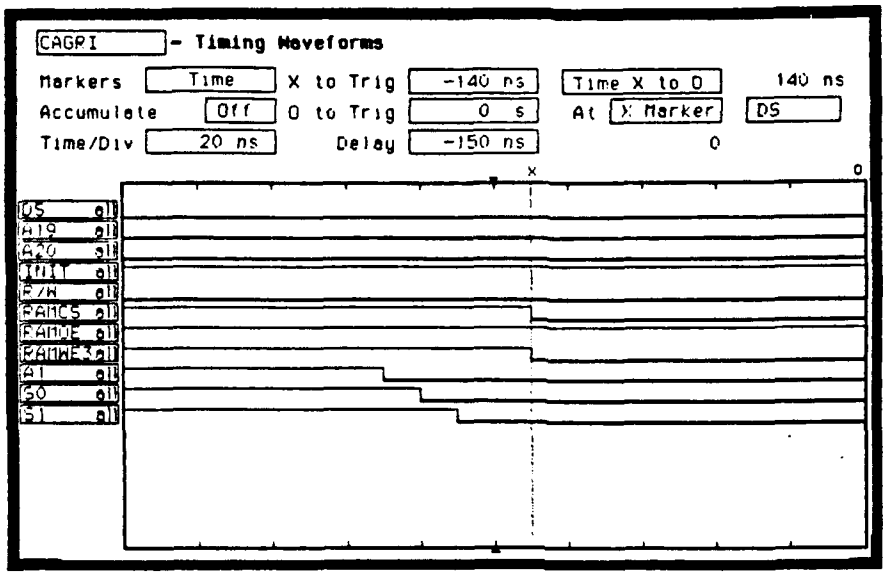


Figure 33. Verification of the second condition of the RAM3 Write Enable Signal

(3) The Third Condition of RAMW'E3 Output. When A20 = 0, A19 = 0, A1 = 0, S1(SIZ1) = 1, S0(SIZ0) = 1, DS = 0, and RW = 0, the RAM3 Write Enable has to be asserted. As we see in Figure 34, the verification is satisfactory.

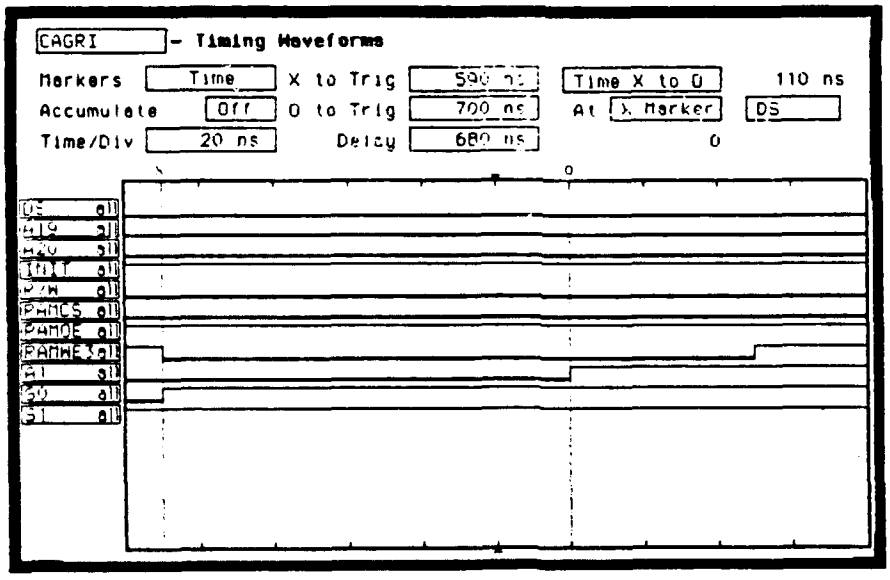


Figure 34. Verification of the third condition of the RAM3 Write Enable Signal

(4) The Fourth Condition of RAMWE3 Output. The fourth condition of the RAMWE3 requires that A20 = 0, A19 = 0, A1 = 0, A0 = 1, S0(SIZ0) = 0, DS = 0, and RW = 0. The verification of this condition is shown the Figure 35.

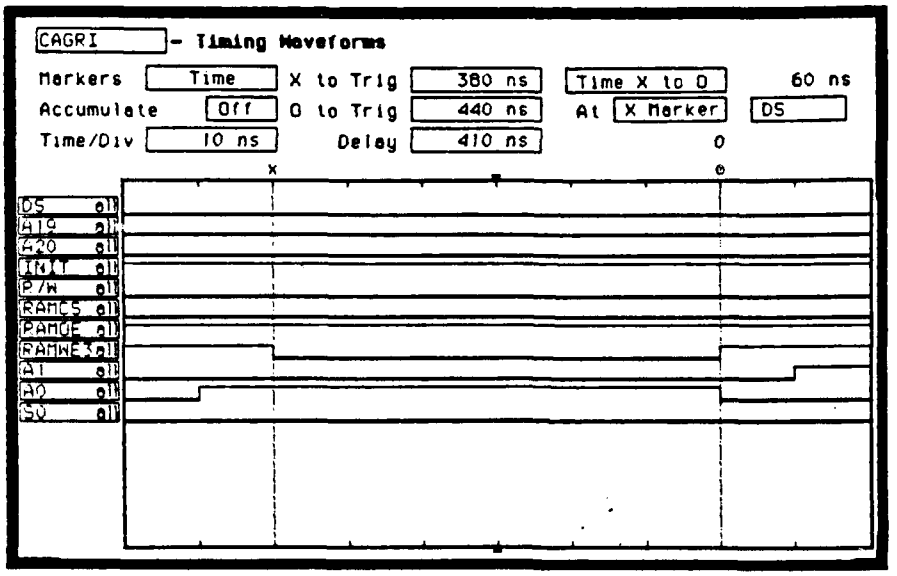


Figure 35. Verification of the fourth condition of the RAM3 Write Enable Signal

g. Verification of RAMWE4 (RAM4 Write Enable Signal) Output

(1) The First Condition of RAMWE4 Output. The first condition of the RAMWE4 requires A20 = 0, A19 = 0, A0 = 1, S1(SIZ1) = 1, S0(SIZ0) = 1, DS = 0, RW = 0 in order to assert the RAM4 Write Enable. The verification of this condition is shown in the Figure 36.

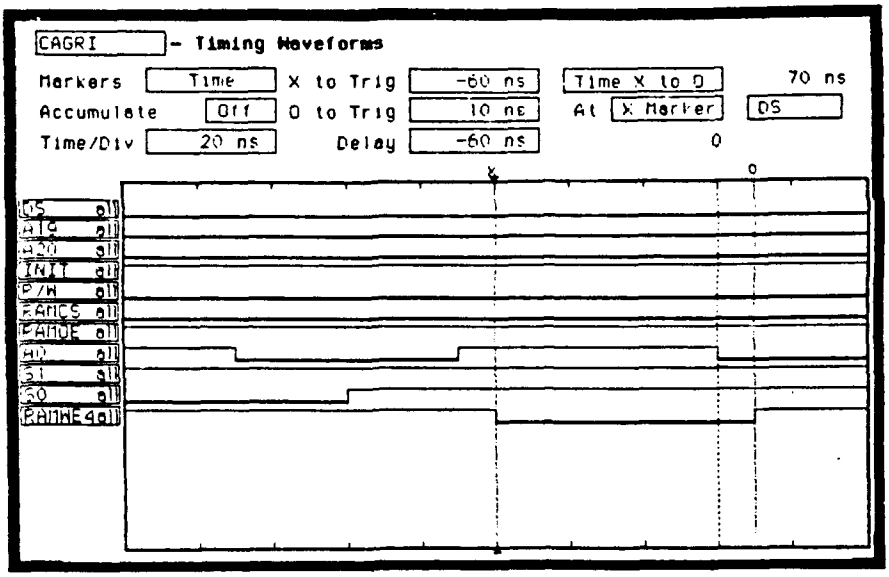


Figure 36. Verification of the first condition of the RAM4 Write Enable signal

(2) The Second Condition of RAMWE4 Output. The second condition of the RAMWE4 requires A20 = 0, A19 = 0, S0(SIZ0) = 0, S1(SIZ1) = 0, DS = 0, RW = 0 in order to assert the RAM4 Write Enable. The verification of this condition is shown in the Figure 37.

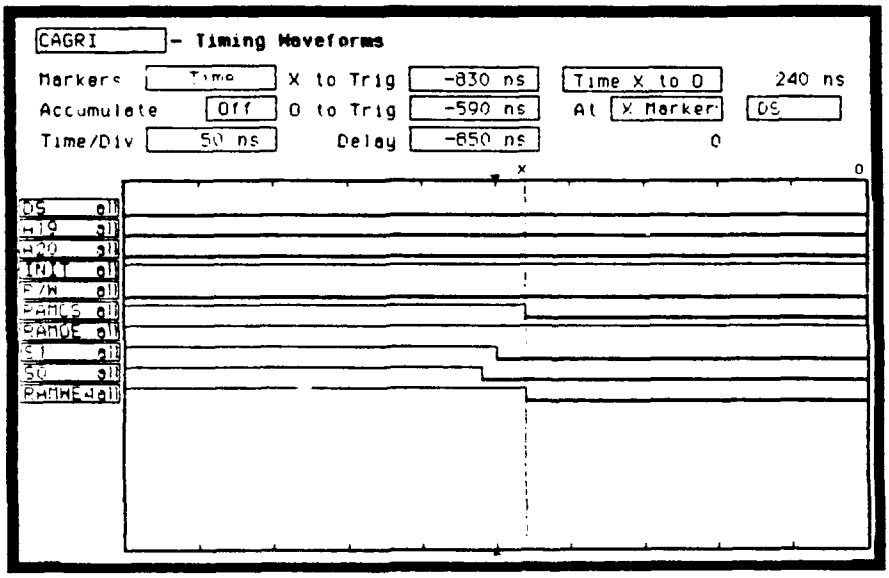


Figure 37. Verification of the second condition of the RAM4 Write Enable Signal

(3) The Third Condition of RAMWE4 Output. The third condition of the RAMWE4 requires A20 = 0, A19 = 0, A0 = 1, A1 = 1, DS = 0, RW = 0 in order to assert the RAM4 Write Enable. The verification of this condition is shown in Figure 38.

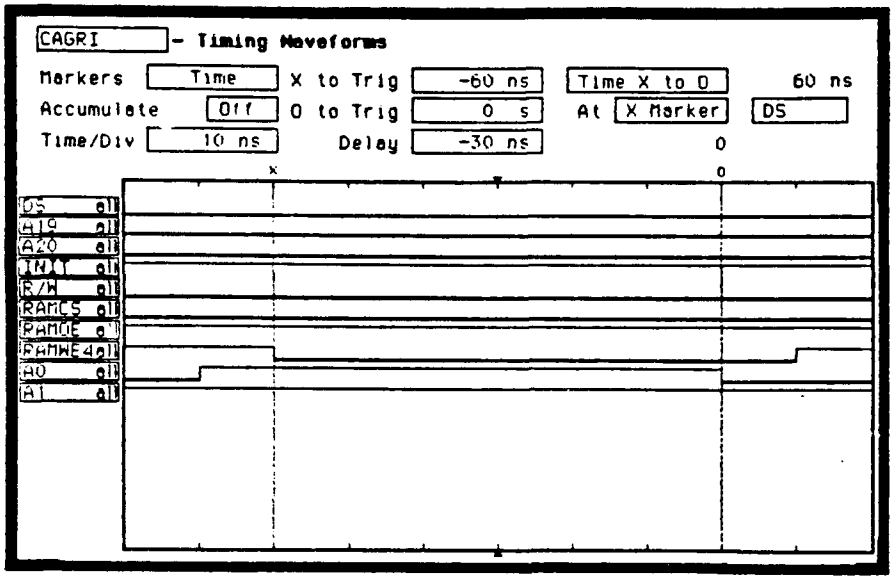


Figure 38. Verification of the third condition of the RAM4 Write Enable Signal

(4) The Fourth Condition of RAMWE4 Output. The fourth condition of the RAMWE4 requires A20 = 0, A19 = 0, A1 = 1, S1(SIZ1) = 1, DS = 0, RW = 0 in order to assert the RAM4 Write Enable. The verification of this condition is shown in the Figure 39.

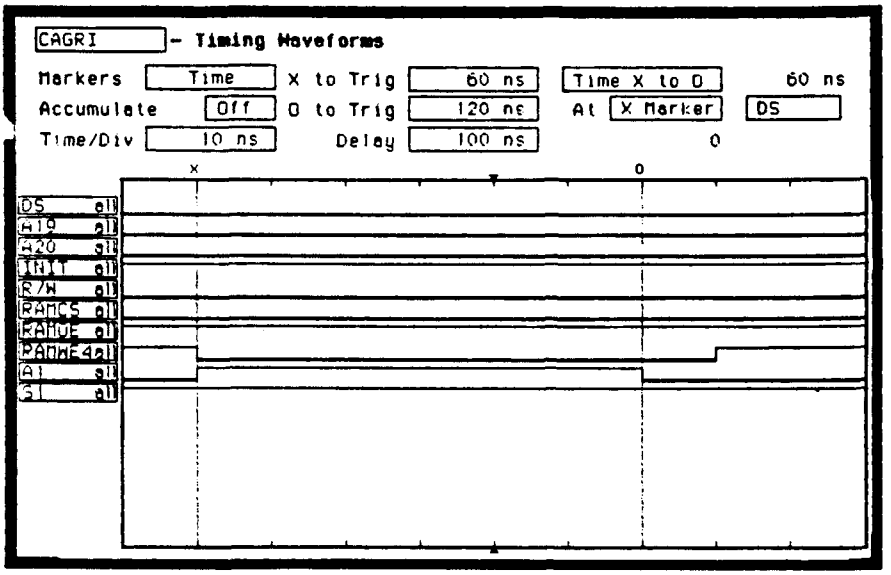


Figure 39. Verification of the fourth condition of the RAM4 Write Enable Signal

h. Verification of ROMCS (ROM Chip Select Signal) Output

(1) The First Condition of ROMCS Output. As a first condition, in order to assert the ROM Chip Select signal it is required that A20 = 0, A19 = 0, INIT = 0, RW = 1, and DS = 0. Figure 40 verifies this condition.

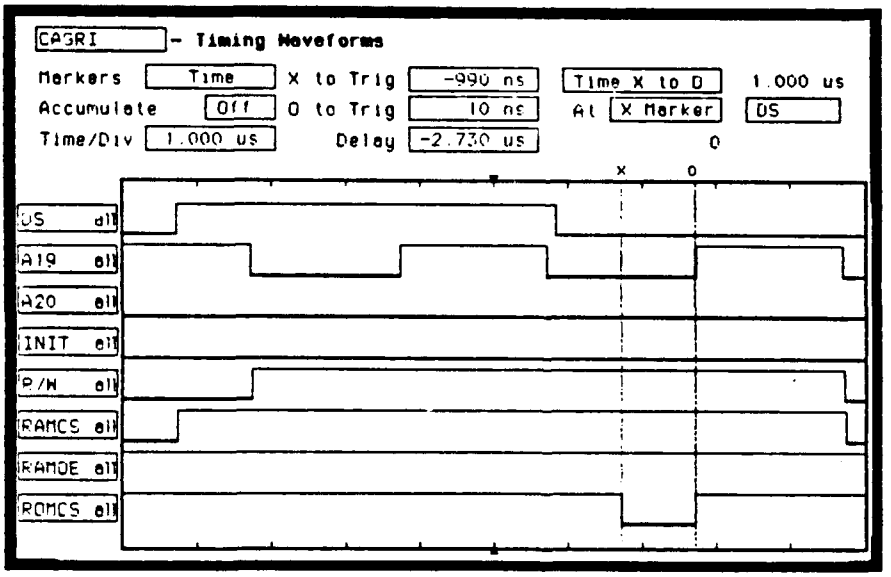


Figure 40. Verification of the first condition of the ROM Chip Select Signal

(2) The Second Condition of ROMCS Output. The second condition, requires A20=0, A19=1, A18=0,INIT=1, RW=1, and DS=0 in order to assert the ROM Chip Select signal. The Figure 41 verifies this condition.

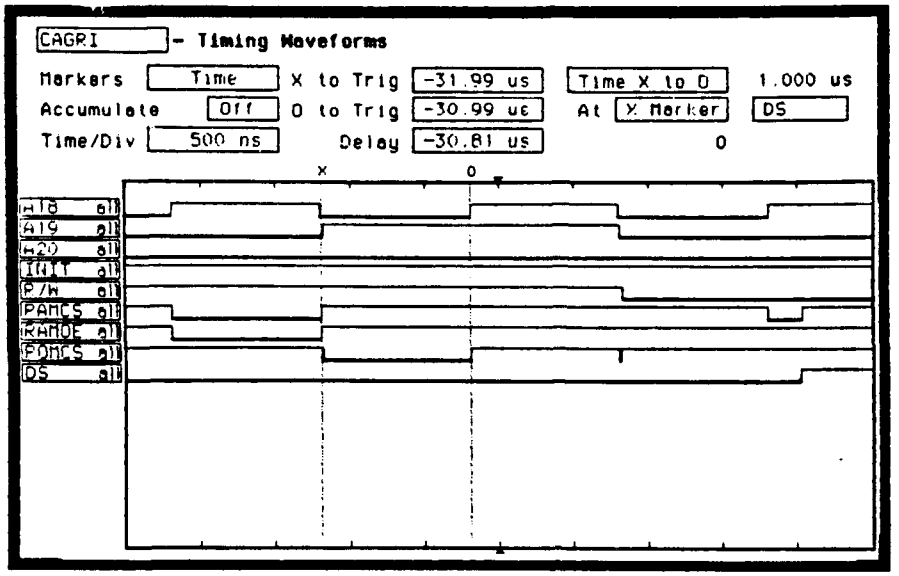


Figure 41. Verification of the second condition of the ROM Chip Select Signal

i. Verification of CHRINHIB (CHR Clock Inhibit Register) Output

The CHR Clock Inhibit Register is in the PAL_CAGRI and makes the synchronization between exterior and text-video horizontal synchronization signals possible. This D-Flip Flop accepts the CHR (Character) clock signal as a clock signal and then the Horizontal synchronization output of the MC6845 enters the D input of this register (see Figure 7). The Horizontal synchronization signal of the exterior video from the video synchronization unit is used as a reset signal. The output of this register goes to the count inhibit input of the Character clock counter. So, if this output is high, there is no longer a clock signal. At that time, the output of this register stays at high until the exterior horizontal synchronization signal goes to low (synchronization pulse asserted). When the exterior signal goes to low, since the condition of "EXTSYNC=0 and CHRINHIB=1" was selected as a reset equation, the output of this register goes to low. This means that the text-video synchronization signal is halted until the exterior one becomes low. Then the counting resumes. As result of this operation, two separate synchronization signals has been synchronized. This situation is portrayed in

Figure 42. The CRTSYN signal and EXTSYN (exterior synchronization) signal at the beginning is out of phase. But when the CHRINH (Character Counter Inhibit) signal is asserted as required from time to time, these two signal catch and lock onto each other.

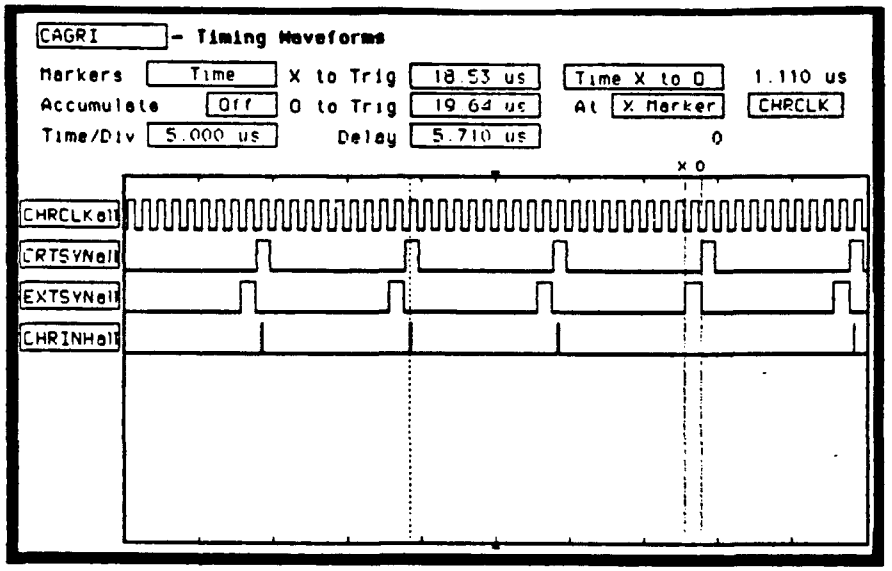


Figure 42. Verification of CHRINH (Character Counter Inhibit) Signal

3. VERIFICATIONS OF THE EPLD-SELINJR

a. Verification of DLY0, DLY6, and Q0-Q3 Outputs

The ENB (Enable) signal used in the CRT controller unit, is made up of 10 clock cycles of 125 ns. For six clock cycles, it is low and for four clock cycles, high. For the generation of the DLY0 and DLY6 signals, the output of a 4-bit delay counter is used. When the counter at zero state DLY0 is high for one clock period. When counter is at the sixth state DLY6 is high for one clock period. By by using these signals, the ENB register in the PAL_SELIN goes high at the sixth clock and after four clock, goes low, when DLY0 goes high. These conditions are clearly shown in Figure 43. The CLKENB signal of EPLD_SELIN is no longer used at this point. The

The Q0-Q3 outputs of this EPLD goes to I0-I3 inputs of EPLD_NESRIN to generate DLY1 and DLY2 signals (see Appendix G).

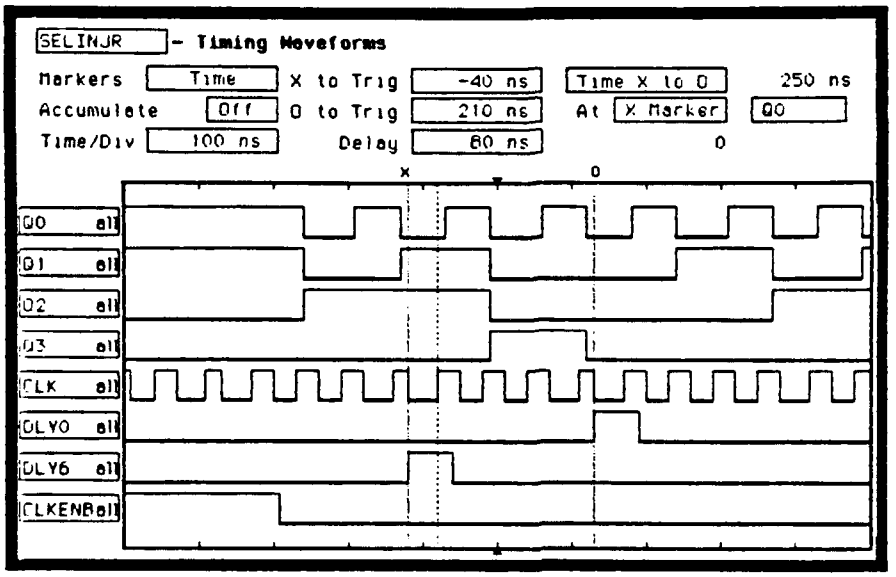


Figure 43. Verification of Delay counter

b. Verification of TEXVI (Text Video) Output

The verification of the CRT Controller combinational circuits during ENBLN (Enable Line), DISEN (Display Enable), and DOTCLK inputs are high, as shown in Figure 44.

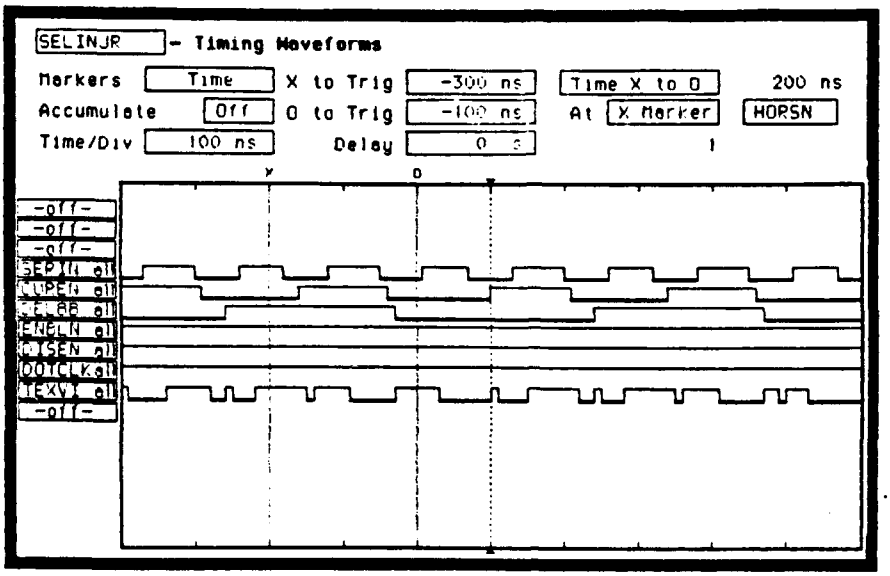


Figure 44. Verification of CRT Controller Combinational Circuit

c. *Verification of TEXSN (Text Video Synchronization) Output*

The horizontal and vertical synchronization outputs of the MC6845 CRT controller chip are ORed with each other in order to generate the composite video signal. The verification of this condition is shown in Figure 45.

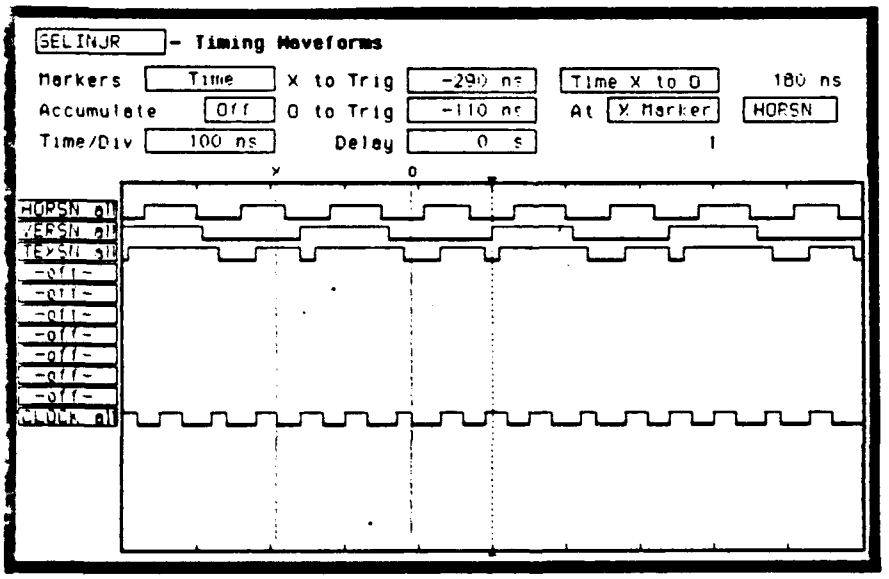


Figure 45. Verification of Synchronization Signal outputs

B. VERIFICATIONS OF THE OTHER UNITS

The other components used in the CPU unit (seen in Figure 3) are the RAM and ROM memories and DUART chip. On the CRT Controller unit (as seen in Figure 5) the connections are straightforward and there is almost no other way to implement this. In both these units, the components or the sub units are too difficult to verify or sometimes impossible, without operating the whole system. Therefore, those were not verified. However, they should have been verified at the time in which the satisfactory system operation was obtained.

The last unit, the "synchronization and multiplexing" unit, has been verified from the very beginning of the design and implementation phase. Actually, it had been designed throughout the satisfactory verifications. The operation of that system has also been demonstrated during the thesis presentation. Any further verification was considered unnecessary.

V. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

In this thesis, a stand-alone microcomputer based on video-text generation was designed and implemented. This system can operate independently without the need for other devices.

This system was primarily designed as an educational microcomputer system with its own video-text output. The system is designed to run at 16 MHz, the maximum clock speed of the MC68020 CPU. Because the system was implemented using "wire wrap" (which can cause "cross talk" between connection lines) the optimum speed estimated was 8 MHz. Once the system is functional nothing is additionally required to run the system at 16 MHz, because the dot clock signal of the CRT Controller is already set at 16 Mhz and the system was designed so that the memory timing allows 16 MHz clock operation. The only thing that cannot be changed is the character clock rate of the CRT Controller, which is 2 Mhz. Its maximum clock speed is 2.5 MHz and this clock rate of 2 MHz was selected during design.

Although all the units and the programmable devices in the system were tested and verified as shown in Chapter IV, a satisfactory system operation could not be obtained since the system software is not yet available. Nevertheless, this system was designed to function with all necessary units and the hardware system was validated.

The analog part of the system works as expected. Using this part (the synchronization and multiplexing unit) the system can place its text-video output onto another NTSC standard video signal.

B. RECOMMENDATIONS AND FUTURE STUDIES

The VTG system was implemented using "wire wrap". The "wire wrap" is preferable for fast implementation of a prototype, however, it causes "cross talk" between the lines and makes it difficult to troubleshoot. Therefore, it is recommended that the VTG be replaced on a PCB (Printed Circuit Board) to eliminate these kinds of problems.

For future study the following improvements may be made to the VTG system :

- A sophisticated keyboard routine to support any kind of complicated keyboard operations.
- The initialization register tables of the CRT Controller placed in a separate ROM as a firmware.
- A versatile communication routine written to communicate with several kind of peripheral such as printers or other microcomputers through the parallel output port of MC68681.
- The light pen feature of the CRT Controller operated and some improved text processing software written to use this feature.
- Adding a second synchronization unit to synchronize the text-video with the color burst signal of the exterior video signal, to obtain a colored video-text-generation.
- Adding the 63484 ACRTC (ADVANCED CRT CONTROLLER) chip to the system, to obtain advanced graphics. This will require MMU firmware modification.

APPENDIX A. MC68020 SIGNAL DESCRIPTION

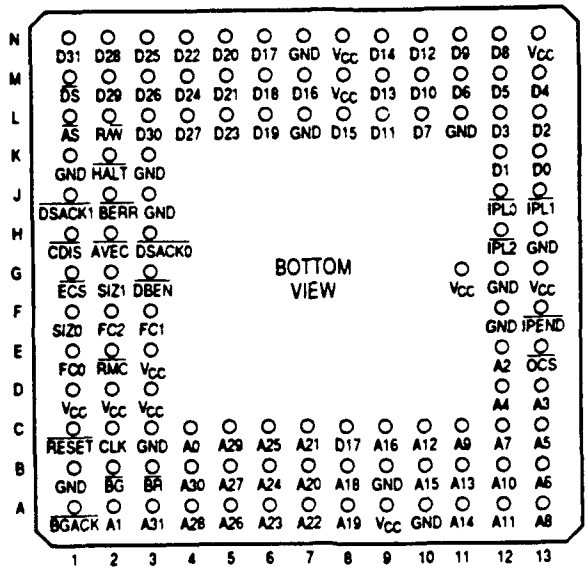


Figure 46. MC68020 Pin Assignment for RC, RL, and RP Suffix (Copied from Reference 2)

A. ADDRESS BUS SIGNALS

(*A*0 – *A*31)

- Address Bus outputs work as 3-state (This feature makes possible the "Direct Memory Access" by DMAC's).
- Address bus provides the addressing up to 4 Gigabyte.
- A6, A18, A19, A20 and A21 outputs are used to generate the chip-select and the chip-control signals on the Video Text Generator.

B. DATA BUS SIGNALS

(*D*0 – *D*31)

- Data Bus inputs/outputs work as 3-state (This feature makes possible the "Direct Memory Access" by DMAC's).
- Data bus provides data exchange between MC68020 and the other devices (like peripheral adaptors, CRT Controllers, Memory units, etc.).
- On the Video Text Generator, D0 through D31 are used for RAMs (62256LP), and D24 through D31 are used for EPROM (27C256), DUART (MC68681), and CRT-CONTROLLER (MC6845) those work only byte-wise.
- Byte-wise Data Transfer is only available through the D24-D31 Data port.

C. TRANSFER SIZE SIGNALS

(*SIZ*0, *SIZ*1)

- These outputs work as 3-state.
- They show us how many bytes of an operand remained to be transferred in a bus cycle, as seen in Table 6.
- On the Video Text Generator these outputs are only used to generate the RAM Write enable signals.

D. ADDRESS STROBE

($\overline{AS}$)

- This output works as 3-state output.
- Informs the availability of valid function code, address and read-write information on the bus.

Table 6. TRANSFER SIZE CODES DESCRIPTION

SIZE1	SIZE0	SIZE
L	H	BYTE
H	L	WORD
H	H	3 BYTES
L	L	LONGWORD

- It is used on the address decoder as a synchronization pulse.

E. DATA STROBE

$(\overline{DS})$

- This output works as 3-state output.
- During the read cycle, it triggers the slave device to drive the data bus. During the write cycle it shows that a valid data is available on the data bus.
- For thesis research this output was used as an input to the ADDRESS-DECODER-PAL in order to generate chip select signals.

F. READ-WRITE

$(R/\overline{W})$

- This output is a 3-state output.
- During the read cycle to an external device, a High level appears on this output. In contrary, during the write cycle, to an external device a Low level appears on this output. If there is neither a write nor read cycle it becomes open (High impedance). This output is therefore used to determine the read or write positions

of the RAM's and to enable the chip select of RAMs during initialization only and ROM on the Video Text Generator.

G. DATA BUFFER ENABLE

($\overline{DBEN}$)

- This output is also a 3-state output.
- When using external data buffers, it operates as a trigger signal.

H. DATA TRANSFER AND SIZE ACKNOWLEDGE

($\overline{DSACK0}$, $\overline{DSACK1}$)

- These are inputs.
- As can be seen in the Table 7, they describe the port size of external device and the completion of the data transfer.
- In this thesis study they describe the 8-bit port size for ROM, DUART, and CRT CONTROLLER and 32-Bit port size for RAMs.

Table 7. DATA TRANSFER SIZE AND COMPLETION SIGNALS DESCRIPTION

DSACK1	DSACK0	RESULT OF OPERATION
1	1	INSERTS WAIT STATES
1	0	COMPLETION WITH 8 BIT PORT SIZE
0	1	COMPLETION WITH 16 BIT PORT SIZE
0	0	COMPLETION WITH 32 BIT PORT SIZE

I. BUS REQUEST

$\overline{(BR)}$

- This is an input.
- It indicates that some other devices request to have the bus.

J. BUS GRANT

$\overline{(BG)}$

- This is an output.
- It describes that 68020 is going to grant the bus to the requesting device upon the completion of the current bus cycle.

K. BUS GRANT ACKNOWLEDGE

$\overline{(BGACK)}$

- This is an input.
- It describes that the requesting device has taken over the bus from the 68020.

L. BUS ERROR

$\overline{(BERR)}$

- This is an input.
- It tells that something is wrong with the current bus cycle.

M. HALT

$\overline{(HALT)}$

- This an open drain input and output.
- If it is used as an output, it signals the external devices that the 68020 has halted. If used as an input, the 68020 is going to be halted, previous bus cycle information

is kept about read/write, function code, size signals and address bus. Meanwhile the data bus stays in High impedance state., and all control signals becomes inactive.

N. READ-MODIFY-WRITE CYCLE

$\overline{(RMC)}$

- This is a three state output.
- It tells that an indivisible read-modify-write cycle is on the bus.

O. EXTERNAL CYCLE START

$\overline{(ECS)}$

- This an output.
- It shows the start of an external bus cycle at any time.

P. OPERAND CYCLE START

$\overline{(OCS)}$

- This is an output.
- It indicates the start of an instruction prefetch or an operand transfer.

Q. CACHE DISABLE

$\overline{(CDIS)}$

- This is an input.
- Disable the on-chip cache memory. If cache memory will not be used this should be pulled down to ground to disable the on-chip cache memory.

R. FUNCTION CODE SIGNALS

$(FC0 - FC2)$

- These are 3-state outputs.

- They describes the processor and address space of the current bus cycle as seen in Table 8 on page 69.

Table 8. FUNCTION CODES DESCRIPTION

FC2	FC1	FC0	ADDRESS SPACE
L	L	L	UNDEFINED
L	L	H	USER DATA SPACE
L	H	L	USER PROGRAM SPACE
L	H	H	UNDEFINED
H	L	L	UNDEFINED
H	L	H	SUPERVISOR DATA SPACE
H	H	L	SUPERVISOR PROGRAM SPACE
H	H	H	CPU SPACE

S. INTERRUPT PRIORITY LEVEL SIGNALS

($\overline{IPL0}$-$\overline{IPL2}$)

- These are inputs.
- These show the level of interrupt requested by an external device, as can be seen in Table 9.

T. INTERRUPT PENDING

($\overline{IPEND}$)

- This is an output.

Table 9. INTERRUPT PRIORITY LEVELS AND MASK VALUES

IP2	IP1	IP0	INTERRUPT LEVEL	INTERRUPT MASK LEVEL
1	1	1	0	N/A
1	1	0	1	0
1	0	1	2	1-0
1	0	0	3	2-0
0	1	1	4	3-0
0	1	0	5	4-0
0	0	1	6	5-0
0	0	0	7	7-0

- This output indicates that the active interrupt priority level is higher than the level of the interrupt mask existing in the status register or tells that the recognition of a nonmaskable (level 7) interrupt.

U. RESET

($\overline{RESET}$)

- This is an open drain input and output.
- If this is used as an input, 68020 goes into reset exception processing; if used as an output, all connected external devices goes into reset and no internal activity happens.
- The reset was used in this thesis as an input to accomplish the hard reset of the system.

V. AUTOVECTOR

($\overline{AVEC}$)

- This is an input.
- When this input is asserted during an interrupt acknowledge cycle, an interrupt vector is internally generated.

W. CLOCK

(*CLK*)

- This an TTL compatible clock input to the 68020.
- The maximum clock frequency to be used depends on the type of the 68020 chip. 68020 used for this thesis has been working with 8 MHz of clock frequency.

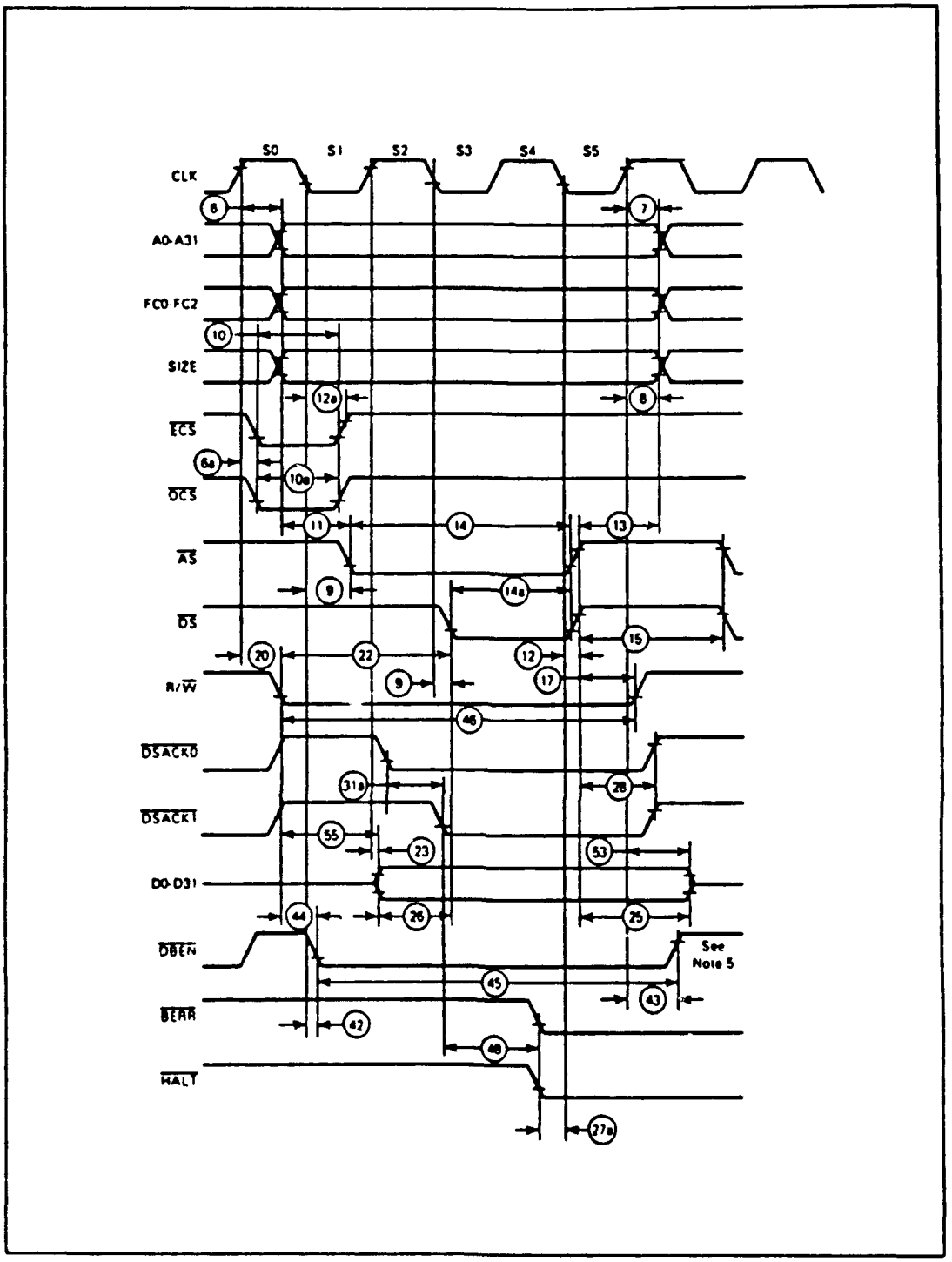


Figure 47. Write Cycle Timing Diagram of MC68020 (Copied from Reference 1)

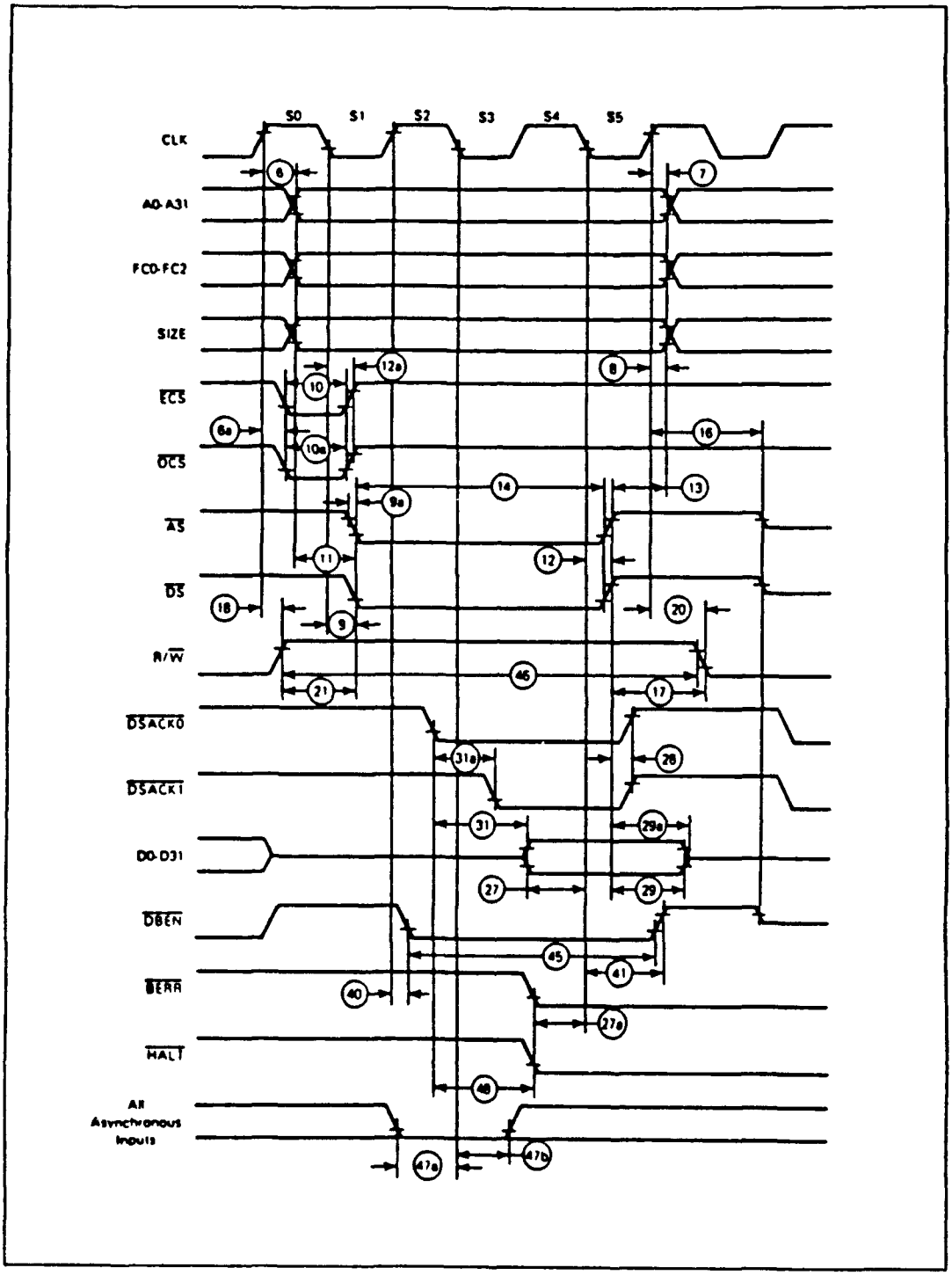


Figure 48. Read Cycle Timing Diagram of MC68020 (Copied from Reference 1)

Table 10. AC ELECTRICAL CHARACTERISTICS OF MC68020 (COPIED FROM REFERENCE 1)

Num	Characteristic	Symbol	MC68020RC12 Min	MC68020RC12 Max	MC68020RC16 Min	MC68020RC16 Max	Unit
6	Clock High to Address, FC, Size, RMC Valid	t_{CHAV}	0	40	0	30	ns
6A	Clock High to ECS, OCS Asserted	t_{CHEV}	0	30	0	20	ns
7	Clock High to Address, Data, FC, RMC, Size High Impedance	t_{CHAZx}	0	80	0	80	ns
8	Clock High to Address/FC, Size, RMC Invalid	t_{CHAZn}	0	–	0	–	ns
9	Clock Low to AS, DS Asserted	t_{CLSA}	3	40	3	30	ns
9A	AS to DS Assertion (Read) (Skew)	t_{STSA}	– 20	20	– 15	15	ns
10	ECS Width Asserted	t_{ECSA}	25	–	20	–	ns
10A	OCS Width Asserted	t_{OCSA}	25	–	20	–	ns
11[6]	Address, FC, Size, RMC Valid to AS (and DS Asserted Read)	t_{AVSA}	20	–	15	–	ns
12	Clock Low to AS, DS Negated	t_{CLSN}	0	40	0	30	ns
12A	Clock Low to ECS, OCS Negated	t_{CLEN}	0	40	0	30	ns
13	AS, DS Negated to Address, FC, Size Invalid	t_{SNAI}	20	–	15	–	ns
14	AS (and DS Read) Width Asserted	t_{SWA}	120	–	100	–	ns
14A	DS Width Asserted Write	t_{SWAW}	50	–	40	–	ns
15	AS, DS Width Negated	t_{SN}	50	–	40	–	ns
16	Clock High to AS, DS, R/W, DBEN High Impedance	t_{CSZ}	–	80	–	60	ns
17[6]	AS, DS Negated to R/W High	t_{SNRN}	20	–	15	–	ns
18	Clock High to R/W High	t_{CHRH}	0	40	0	30	ns
20	Clock High to R/W Low	t_{CHRL}	0	40	0	30	ns
21[6]	R/W High to AS Asserted	t_{RAAA}	20	–	15	–	ns
22[6]	R/W Low to DS Asserted (Write)	t_{RASA}	90	–	70	–	ns
23	Clock High to Data Out Valid	t_{CHDO}	–	40	–	30	ns
25[5]	DS Negated to Data Out Invalid	t_{SNDI}	20	–	15	–	ns
26[5]	Data Out Valid to DS Asserted (Write)	t_{DVSA}	20	–	15	–	ns
27	Data In Valid to Clock Low (Data Setup)	t_{DICL}	10	–	5	–	ns
27A	Late BERR, HALT Asserted to Clock Low Setup Time	t_{BELCL}	25	–	20	–	ns
28	AS, DS Negated to DSACKx, BERR, HALT, AVEC Negated	t_{SNDN}	0	110	0	80	ns
29	DS Negated to Data In Invalid (Data In Hold Time)	t_{SNDI}	0	–	0	–	ns
29A	DS Negated to Data In (High Impedance)	t_{SNDI}	–	80	–	60	ns
31[2]	DSACKx Asserted to Data In Valid	t_{DADI}	–	60	–	50	ns
31A[3]	DSACKx Asserted to DSACKx Valid (tDSACK Asserted Skew)	t_{DADV}	–	20	–	15	ns
32	RESET Input Transition Time	t_{HRI}	–	2.5	–	2.5	Clk Per
33	Clock Low to BG Asserted	t_{CLBA}	0	40	0	30	ns
34	Clock Low to BG Negated	t_{CLBN}	0	40	0	30	ns
35	BR Asserted to BG Asserted (RMC Not Asserted)	t_{BRAGA}	1.5	3.5	1.5	3.5	Clk Per
37	BGACK Asserted to BG Negated	t_{GAGN}	1.5	3.5	1.5	3.5	Clk Per
39	BG Width Negated	t_{GN}	120	–	80	–	ns
39A	BG Width Asserted	t_{GA}	120	–	80	–	ns
40	Clock High to DBEN Asserted (Read)	t_{CHDAR}	0	40	0	30	ns
41	Clock Low to DBEN Negated (Read)	t_{CLDNR}	0	40	0	30	ns
42	Clock Low to DBEN Asserted (Write)	t_{CLDAW}	0	40	0	30	ns
43	Clock High to DBEN Negated (Write)	t_{CHDNW}	0	40	0	30	ns
44[6]	R/W Low to DBEN Asserted (Write)	t_{RADA}	20	–	15	–	ns
45[5]	DBEN Width Asserted — Read	t_{DA}	80	–	60	–	ns
	DBEN Width Asserted — Write		160	–	120	–	ns
46	R/W Width Asserted (Write or Read)	t_{RWA}	180	–	150	–	ns
47a	Asynchronous Input Setup Time	t_{AIST}	10	–	5	–	ns
47b	Asynchronous Input Hold Time	t_{AIHT}	20	–	15	–	ns
48[4]	DSACKx Asserted to BERR, HALT Asserted	t_{DABA}	–	35	–	30	ns
53	Data Out Hold from Clock High	t_{DOCH}	0	–	0	–	ns
55	R/W Asserted to Data Bus Impedance Change	t_{RADC}	40	–	40	–	ns
56	RESET Pulse Width (Reset Instruction)	t_{HRPW}	512	–	512	–	Clks
57	BERR Negated to HALT Negated (Rerun)	t_{BNHN}	0	–	0	–	ns

NOTES

1 This number can be reduced to 5 nanoseconds if strobes have equal loads.

2 If the asynchronous setup time (#47) requirements are satisfied, the DSACKx low to data setup time (#31) and DSACKx low to BERR low setup time (#48) can be ignored. The data must only satisfy the data-in to clock low setup time (#27) for the following clock cycle. BERR must only satisfy the late BERR low to clock low setup time (#27A) for the following clock cycle.

3 This parameter specifies the maximum allowable skew between DSACK0 to DSACK1 asserted or DSACK to DSACK0 asserted. Specification #47 must be met by DSACK0 or DSACK1.

4 In the absence of DSACKx, BERR is an asynchronous input using the asynchronous input setup time (#47).

5 DBEN may stay asserted on consecutive write cycles.

6 Actual value depends on the clock input waveform.

Table 11. MC68020 INSTRUCTION SET (COPIED FROM REFERENCE 1)

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
ADDA	Add Address
ADDI	Add Immediate
ADDQ	Add Quick
ADDX	Add with Extend
AND	Logical AND
ANDI	Logical AND Immediate
ASL, ASR	Arithmetic Shift Left and Right
Bcc	Branch Conditionally
BCHG	Test Bit and Change
BCLR	Test Bit and Clear
BFCHG	Test Bit Field and Change
BFCLR	Test Bit Field and Clear
BFEXTS	Signed Bit Field Extract
BFEXTU	Unsigned Bit Field Extract
BFFFO	Bit Field Find First One
BFINS	Bit Field Insert
BFSET	Test Bit Field and Set
BFTST	Test Bit Field
BKPT	Breakpoint
BRA	Branch
BSET	Test Bit and Set
BSR	Branch to Subroutine
BTST	Test Bit
CALLM	Call Module
CAS	Compare and Swap Operands
CAS2	Compare and Swap Dual Operands
CHK	Check Register Against Bound
CHK2	Check Register Against Upper and Lower Bounds
CLR	Clear
CMP	Compare
CMPA	Compare Address
CMPI	Compare Immediate
CMPM	Compare Memory to Memory
CMP2	Compare Register Against Upper and Lower Bounds
DBcc	Test Condition, Decrement and Branch
DIVS, DIVSL	Signed Divide
DIVU, DIVUL	Unsigned Divide
EOR	Logical Exclusive OR
EORI	Logical Exclusive OR Immediate
EXG	Exchange Registers
EXT, EXTB	Sign Extend
ILLEGAL	Take Illegal Instruction Trap
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link and Allocate
LSL, LSR	Logical Shift Left and Right
MOVE	Move
MOVEA	Move Address
MOVE CCR	Move Condition Code Register
MOVE SR	Move Status Register
MOVE USP	Move User Stack Pointer
MOVEC	Move Control Register
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral
MOVEQ	Move Quick
MOVES	Move Alternate Address Space
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NEGX	Negate with Extend
NOP	No Operation
NOT	Logical Complement
OR	Logical Inclusive OR
ORI	Logical Inclusive OR Immediate
ORI CCR	Logical Inclusive OR Immediate to Condition Codes
ORI SR	Logical Inclusive OR Immediate to Status Register
PACK	Pack BCD
PEA	Push Effective Address
RESET	Reset External Devices
ROL, ROR	Rotate Left and Right
ROXL, ROXR	Rotate with Extend Left and Right
RTD	Return and Deallocate
RTE	Return from Exception
RTM	Return from Module
RTR	Return and Restore Codes
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
Scc	Set Conditionally
STOP	Stop
SUB	Subtract
SUBA	Subtract Address
SUBI	Subtract Immediate
SUBQ	Subtract Quick
SUBX	Subtract with Extend
SWAP	Swap Register Words
TAS	Test Operand and Set
TRAP	Trap
TRAPcc	Trap Conditionally
TRAPV	Trap on Overflow
TST	Test Operand
UNLK	Unlink
UNPK	Unpack BCD

COPROCESSOR INSTRUCTIONS

Mnemonic	Description
cpBcc	Branch Conditionally
cpDBcc	Test Coprocessor Condition, Decrement and Branch
cpGEN	Coprocessor General Instruction

Mnemonic	Description
cpRESTORE	Restore Internal State of Coprocessor
cpSAVE	Save Internal State of Coprocessor
cpScc	Set Conditionally
cpTRAPcc	Trap Conditionally

APPENDIX B. MC6845 SIGNAL DESCRIPTION

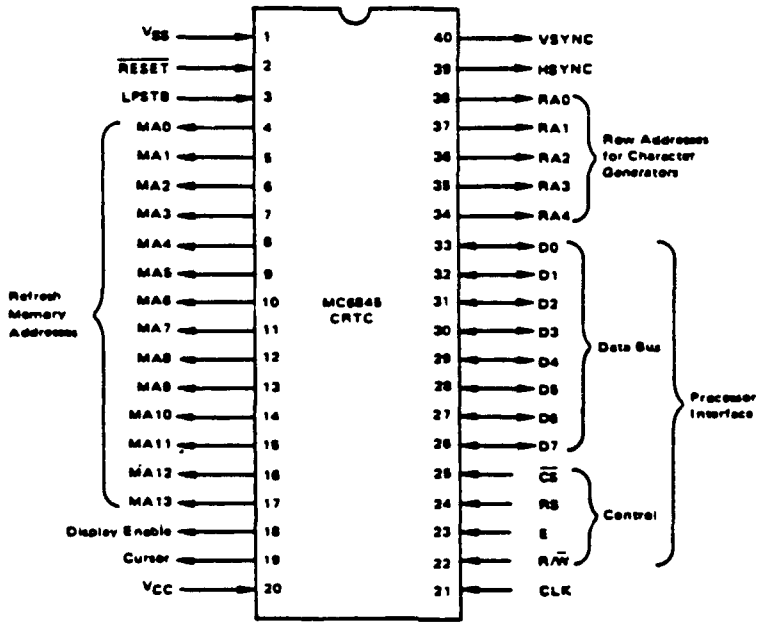


Figure 49. MC6845 Pin Assignment for P Suffix (Copied from Reference 6)

A. DATA BUS SIGNALS

($D0 - D7$)

- Data Bus works bidirectional and 3-state.
- These buffered output/inputs are always high impedance, except the read operations by CPU.

B. ENABLE SIGNAL

(E)

- This is a high impedance input.
- Enables data bus I/O buffers and clocks the data to and from CRT Controller.
- The high to low transitions of this input signal is the active edge.

C. CHIP SELECT SIGNAL

($\overline{CS}$)

- This is a high impedance input.
- When it is low, selects the CRT Controller to read or write to its internal registers.

D. REGISTER SELECT SIGNAL

(RS)

- This is a high impedance input.
- When this input is low, the address register of the internal registers are selected; when it is high, the selected internal register of the address register is accessed.
- On the VTG this input is connected to the A2 address line.

E. READ-WRITE SIGNAL

($R/\overline{W}$)

- This is a high impedance input.
- When this input is low, the internal registers get written, or when high, read.

F. VERTICAL SYNC. SIGNAL

(VSYNC)

- This is an active high signal and determines the start of a new picture frame.

G. HORIZONTAL SYNC. SIGNAL

(*HSYNC*)

- This is an active high signal and determines the start of a new horizontal scan line.

H. DISPLAY ENABLE

(*Display Enable*)

- This is an active high output and determines the displayed area on the video screen.

I. REFRESH MEMORY ADDRESS SIGNALS

(*MA*0 – *MA*13)

- These are outputs.
- These outputs are used to refresh the CTR screen with the pages of data placed into the Memory Refresh RAM by CPU.

J. RASTER ADDRESS SIGNALS

(*RA*0 – *RA*4)

- These are outputs of the internal Raster counter.
- These outputs determine the addresses of the rows for each character for the Character-Generator-ROM.

K. CURSOR SIGNAL

(*CURSOR*)

- This is an active high output.
- This output is used to generate the cursor signal on the video screen.

L. CLOCK SIGNAL

(CLK)

- This is an TTL/MOS compatible clock input.
- High to low transition is the active edge
- This signal is derivated from the dot clock signal.
- Maximum frequency allowed is 2.5 MHz.

M. RESET SIGNAL

$\overline{(RESET)}$

- This is an active low input.
- When this input is low, all the outputs go low, all the counters are reset and stopped but the contents of the control registers do not change.
- In order to reset the MC6845, the Light Pen input has to be low.

N. LIGHT PEN STROBE SIGNAL

(LPSTR)

- This high impedance input latches the current Memory Refresh Address. By doing this the optically sensed position of any screen character or cursor is acknowledged to the CRT controller. So, an optical interaction with the screen is obtained.
- The active edge of this signal is low to high transition.

APPENDIX C. MC68681 SIGNAL DESCRIPTION

In addition to the following descriptions, a signal summary of MC68681 is also included at the end of this Appendix in Table 15.

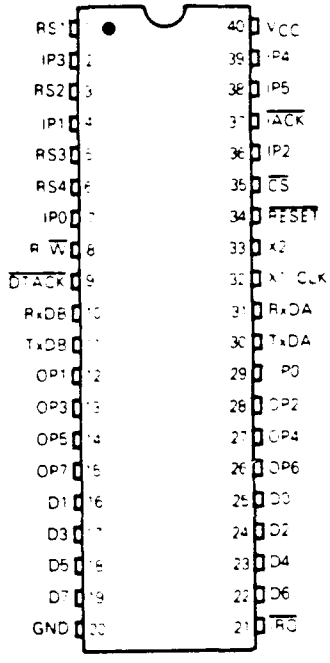


Figure 50. MC68681 Pin Assignment for P, and L Suff.(Copied from Reference 7)

A. CRYSTAL INPUT OR EXTERNAL CLOCK SIGNAL

(X1/CLK)

- This is one of two connection to the crystal used for the internal oscillator or external clock input.

B. CRYSTAL INPUT

($X2$)

- This is the other input connection to the crystal.
- If an exterior clock signal is used, this pin has to be grounded.

C. CHIP SELECT SIGNAL

($\overline{CS}$)

- This is an active low input signal.
- When this signal is low, data transfer is accomplished through the D0-D7 data port under the control of R/W signal and RS1 through RS4 register-select inputs.

D. READ-WRITE

($R/\overline{W}$)

- This is an input.
- When this signal is high, a read operation is done by CPU, and if low, a write operation is done to this chip by the CPU.

E. DATA TRANSFER ACKNOWLEDGE SIGNAL

($\overline{DTACK}$)

- This is an usual DTACK output.
- This three-state output acknowledges the bus master at the completion of bus cycle.

F. REGISTER SELECT BUS SIGNALS

($RS1 - RS4$)

- These inputs are used to select the addresses of the internal registers.
- On the VTG, these inputs are connected to the A2-A5 address lines of the CPU.

- The Register Address Selections are shown on Table 12 on page 82.

Table 12. SELECTING THE REGISTER ADDRESSES AND ADDRESS TRIGGERED COMMANDS (COPIED FROM REFERENCE 7)

RS4	RS3	RS2	RS1	Read (R/$\overline{W}$ = 1)		Write (R/$\overline{W}$ = 0)	
0	0	0	0	Mode Register A	(MR1A, MR2A)	Mode Register A	(MR1A, MR2A)
0	0	0	1	Status Register A	(SRA)	Clock-Select Register A	(CSRA)
0	0	1	0	Do Not Access*		Command Register A	(CRA)
0	0	1	1	Receiver Buffer A	(RBA)	Transmitter Buffer A	(TBA)
0	1	0	0	Input Port Change Register	(IPCR)	Auxiliary Control Register	(ACR)
0	1	0	1	Interrupt Status Register	(ISR)	Interrupt Mask Register	(IMR)
0	1	1	0	Counter Mode: Current MSB of Counter	(CUR)	Counter/Timer Upper Register	(CTUR)
0	1	1	1	Counter Mode: Current LSB of Counter	(CLR)	Counter/Timer Lower Register	(CTLR)
1	0	0	0	Mode Register B	(MR1B, MR2B)	Mode Register B	(MR1B, MR2B)
1	0	0	1	Status Register B	(SRB)	Clock-Select Register B	(CSRB)
1	0	1	0	Do Not Access*		Command Register B	(CRB)
1	0	1	1	Receiver Buffer B	(RBB)	Transmitter Buffer B	(TBB)
1	1	0	0	Interrupt-Vector Register	(IVR)	Interrupt-Vector Register	(IVR)
1	1	0	1	Input Port (Unlatched)		Output Port Configuration Register	(OPCR)
1	1	1	0	Start-Counter Command**		Output Port Register (OPR) / Bit Set Command**	
1	1	1	1	Stop-Counter Command**		Output Port Register (OPR) / Bit Reset Command**	

*This address location is used for factory testing of the DUART and should not be read. Reading this location will result in undesired effects and possible incorrect transmission or reception of characters. Register contents may also be changed.

**Address triggered commands

G. DATA BUS SIGNALS

($D0 - D7$)

- These are bi-directional three-state outputs/inputs.
- These are used for the data transfer between the CPU and DUART.

H. INTERRUPT REQUEST SIGNAL

($\overline{IRQ}$)

- This is an active low open-drain output.
- With this signal output, CPU is acknowledges that one or more of eight maskable interrupting conditions are true [Ref.7: p. (2-3)].

I. CHANNEL A TRANSMITTER SERIAL-DATA OUTPUT

(TxDA)

- This is the serial-data transmitting output for channel A.
- Data is shifted from this pin on the falling edge of the programmed clock cycle.
- The least significant bit is sent first.

J. CHANNEL A RECEIVER SERIAL-DATA INPUT

(RxDA)

- This is the serial-data receiving input for channel A.
- Data is sampled on the rising edge of the programmed clock source [Ref.7: p. (2-3)].
- The least significant bit is received first.

K. CHANNEL B TRANSMITTER SERIAL-DATA OUTPUT

(TxDB)

- This is the serial-data transmitting output for channel B.
- Data is shifted from this pin on the falling edge of the programmed clock cycle.
- The least significant bit is sent first.

L. CHANNEL B RECEIVER SERIAL-DATA INPUT

(RxDB)

- This is the serial-data receiving input for channel B.
- Data is sampled on the rising edge of the programmed clock source [Ref.7: p. (2-3)].
- The least significant bit is received first.

M. PARALLEL INPUTS

(IP0 – IP5)

- These inputs can be used as general-purpose inputs.
- In addition to the usage above, each input can be programmed to perform an alternate function. The programming of these input pins is given on Table 13.

Table 13. PROGRAMMING THE INPUT PORT FUNCTIONS OF MC68681 (COPIED FROM REFERENCE 7)

Function	Input Port Pin					
	IP5	IP4	IP3	IP2	IP1	IP0
General Purpose	Default	Default	Default	Default	Default	Default
Change-of-State Detector			Default	Default	Default	Default
External Counter 1X Clock Input				ACR[6 4]* = 000		
External Timer 16X Clock Input				ACR[6 4]* = 100		
External Timer 1X Clock Input				ACR[6 4]* = 101		
RxCA 16X		CSRA[7 4] = 1110				
RxCA 1X		CSRA[7 4] = 1111				
TxCA 16X			CSRA[3 0] = 1110			
TxCA 1X			CSRA[3 0] = 1111			
RxCB 16X				CSRB[7 4] = 1110		
RxCB 1X				CSRB[7 4] = 1111		
TxCB 16X	CSRB[3 0] = 1110					
TxCB 1X	CSRB[3 0] = 1111					
TxCTSA						MR2A[4] = 1
TxCTSB					MR2B[4] = 1	

NOTE Default refers to the function the input port pins perform when not used in one of the other modes. Only those functions which show the register programming are available for use.

*In these modes, because IP2 is used for the counter/timer-clock input, it is not available for use as the channel B receiver-clock input.

N. PARALLEL OUTPUTS

(OP0 – IP7)

- These outputs can be used as general-purpose outputs.

- In addition to the usage above, each output can be programmed to perform an alternate function. The programming of these output pins is given on Table 14 on page 85.

- On the VTG, the OP4 Receiver Ready/Buffer Full output was used to acknowledge the keyboard unit for handshaking.

Table 14. PROGRAMMING THE OUTPUT PORT FUNCTIONS OF MC68681 (COPIED FROM REFERENCE 7)

Function	Output Port Pin							
	OP7	OP6	OP5	OP4	OP3	OP2	OP1	OP0
General Purpose	OPCR[7] = 0	OPCR[6] = 0	OPCR[5] = 0	OPCR[4] = 0	OPCR[3 2] = 00	OPCR[1 0] = 00	MR1B[7] = 0 MR2B[5] = 0	MR1A[7] = 0 MR2A[5] = 0
$\overline{\text{CTRDY}}$					OPCR[3 2] = 01, ACR[6] = 0*			
Timer Output					OPCR[3 2] = 01, ACR[6] = 1*			
TxCB 1X					OPCR[3 2] = 10			
RxCB 1X					OPCR[3 2] = 11			
TxCA 16X						OPCR[1 0] = 01		
TxCA 1X						OPCR[1 0] = 10		
RxCA 1X						OPCR[1 0] = 11		
$\overline{\text{TxRDYA}}$		OPCR[6] = 1*						
$\overline{\text{TxRDYB}}$	OPCR[7] = 1*							
$\overline{\text{RxRDYA}}$				OPCR[4] = 1, MR1A[6] = 0*				
$\overline{\text{RxRDYB}}$			OPCR[5] = 1, MR1B[6] = 0*					
$\overline{\text{FFULLA}}$				OPCR[4] = 1, MR1A[6] = *				
$\overline{\text{FFULLB}}$			OPCR[5] = 1, MR1B[6] = 1*					
$\overline{\text{RxRTSA}}$								MR1A[7] = 1
$\overline{\text{TxRTSA}}$								MR2A[5] = 1
$\overline{\text{RxRTSB}}$							MR1B[7] = 1	
$\overline{\text{TxRTSB}}$							MR2B[5] = 1	

Note Only those functions which show the register programming are available for use
* Pin requires a pullup resistor if used for this function

O. RESET

$\overline{(RESET)}$

- This is an active low input signal.

- When RESET is low, the input registers of MC68681 are cleared and the interrupt vector register is initialized. Parallel outputs are placed in high state, counter/timer is put in timer mode and channel A and B is inactivated.

Table 15. THE SIGNAL SUMMARY OF THE MC68681 (COPIED FROM REFERENCE 7)

Signal Name	Mnemonic	Pin No.	Input/Output	Active State
Power Supply (+5 V)	V_{CC}	40	Input	High
Ground	GND	20	Input	Low
Crystal Input or External Clock	X1/CLK	32	Input	–
Crystal Input	X2	33	Input	–
Reset	$\overline{RESET}$	34	Input	Low
Chip Select	$\overline{CS}$	35	Input	Low
Read/Write	R/$\overline{W}$	8	Input	High/Low
Data Transfer Acknowledge	$\overline{DTACK}$	9	Output*	Low
Register-Select Bus Bit 4	RS4	6	Input	High
Register-Select Bus Bit 3	RS3	5	Input	High
Register-Select Bus Bit 2	RS2	3	Input	High
Register-Select Bus Bit 1	RS1	1	Input	High
Bidirectional-Data Bus Bit 7	D7	19	Input/Output	High
Bidirectional-Data Bus Bit 6	D6	22	Input/Output	High
Bidirectional-Data Bus Bit 5	D5	18	Input/Output	High
Bidirectional-Data Bus Bit 4	D4	23	Input/Output	High
Bidirectional-Data Bus Bit 3	D3	17	Input/Output	High
Bidirectional-Data Bus Bit 2	D2	24	Input/Output	High
Bidirectional-Data Bus Bit 1	D1	16	Input/Output	High
Bidirectional-Data Bus Bit 0 (Least-Significant Bit)	D0	25	Input/Output	High
Interrupt Request	$\overline{IRQ}$	21	Output*	Low
Interrupt Acknowledge	$\overline{IACK}$	37	Input	Low
Channel A Transmitter Serial Data	TxDA	30	Output	–
Channel A Receiver Serial Data	RxDA	31	Input	–
Channel B Transmitter Serial Data	TxDB	11	Output	–
Channel B Receiver Serial Data	RxDB	10	Input	–
Parallel Input 5	IP5	36	Input	–
Parallel Input 4	IP4	39	Input	–
Parallel Input 3	IP3	2	Input	–
Parallel Input 2	IP2	38	Input	–
Parallel Input 1	IP1	4	Input	–
Parallel Input 0	IP0	7	Input	–
Parallel Output 7	OP7	15	Output**	–
Parallel Output 6	OP6	26	Output**	–
Parallel Output 5	OP5	14	Output**	–
Parallel Output 4	OP4	27	Output**	–
Parallel Output 3	OP3	13	Output**	–
Parallel Output 2	OP2	28	Output	–
Parallel Output 1	OP1	12	Output	–
Parallel Output 0	OP0	29	Output	–

APPENDIX D. PAL SELIN PROGRAM FILES

A. PAL SELIN.ABL FILE

```
module PAL_SELIN;
flag '-r3','-f';
title 'PAL ADDRESS DECODER1 AND CHIP CONTROLLER PROGRAMMING FOR USING IN PAL
22V10 FOR THESIS RESEARCH                                          JAN 17 91';

    SELIN  DEVICE  'P22V10';

    CLK                                 PIN 1;
    A6,A20,A21,AS,DS,RW                 PIN 2,3,4,5,6,7;
    DUDTACK,RESET,DLY0,DLY6             PIN 8,9,10,11;
    ROMCS,DSACK1,DSACK0,INHIB           PIN 13,14,15,18;
    MUXEN,CRTWE,CRTCS,DUDWE,DUDCS       PIN 16,20,21,22,23;
    MUXEN,CRTCS,DUDCS                   IsType 'pos,com,feed_pin';
    ENB,INIT                            PIN 17,19;
    ENB,INIT                            IsType 'pos,reg_D,feed_pin';
    CK,H,L,X = .C.,1,0,.X.;

"STATES OF ENABLE COUNTER

        S0 = ^B0 ;
        S1 = ^B1 ;

STATE_DIAGRAM [ENB];

        STATE S0:IF DLY6 THEN S1 ELSE S0;  "0 DURING 6 CLOCK CYCLE
        STATE S1:IF DLY0 THEN S0 ELSE S1;  "1 DURING 4 CLOCK CYCLE

equations
        INHIB = !AS & DLY0;"IF EQUATION IS TRUE INHIBIT THE ENABLE COUNTER"
                           "AND WAIT UNTIL AS=1"
        INIT := (INIT & RESET) # (!DUDCS);
        !DSACK0 =((!AS & ROMCS & CRTCS & DUDCS & !A20)
                  # (!ROMCS & DLY6) # (!CRTCS & ENB)
                  # (!DUDCS & !DUDTACK) # (!MUXEN));
        !DSACK1 = (!AS  & ROMCS & CRTCS & !A20 & DUDCS);
        !MUXEN  = (!A21 & A20   & !ENB  & !AS);"CRT CONTROLLER MUX ENABLE"

        !DUDCS  = (A21 & A6 & !AS);      "68681-DUART CHIP SELECT"
        !DUDWE  = (A21 & A6 & !DS & !RW);"68681-DUART WRITE ENABLE"

        !CRTCS  = (A21 & !A6 & !AS);       "6845 CRT CONTROLLER CHIP SELECT"
        !CRTWE  = (A21 & !A6 & !DS & !RW); "6845 CRT CONTROLLER WRITE ENABLE"

                    "TEST VECTORS OF INHIB OUT
test_vectors ([AS,DLY0] -> [INHIB])
"            ====================
             [L,    L] -> [L];        "ENABLE COUNTER COUNTS
             [H,    L] -> [L];        "ENABLE COUNTER COUNTS
             [H,    H] -> [L];        "ENABLE COUNTER COUNTS
             [L,    H] -> [H];        "ENABLE COUNTER HALTED
```

```
                        "TEST VECTORS OF ENABLE COUNTER
test_vectors ([CLK,DLY0,DLY6,ENB] -> [ENB])
"             ============================
              [CK,  H,   L,   H]  -> [L];
              [CK,  L,   L,   L]  -> [L];
              [CK,  H,   L,   L]  -> [L];
              [CK,  L,   H,   L]  -> [H];
              [CK,  L,   L,   H]  -> [H];
              [CK,  L,   L,   H]  -> [H];


                        "TEST VECTORS OF MUXEN OUT
test_vectors ([CLK,DLY0,DLY6,A21,A20,AS,ENB] -> [MUXEN])
"          C  D  D  A  A  A  E      M
"          L  L  L  2  2  S  N      U
"          K  Y  Y  1  0     B      X
"             0  6                  N
"        ============================
         [CK,H, L, L, H, L, H] -> [L];  "CRT CONTROLLER MUX ENABLE ASSERTED
         [CK,H, L, L, L, L, L] -> [H];  "                           NEGATED
         [CK,H, L, L, H, H, L] -> [H];  "                           NEGATED
         [CK,H, L, L, L, H, L] -> [H];  "                           NEGATED
         [CK,L, H, L, H, L, L] -> [H];  "                           NEGATED
         [L, L, H, L, L, L, H] -> [H];  "                           NEGATED
         [L, L, H, L, H, H, H] -> [H];  "                           NEGATED
         [L, L, H, L, L, H, H] -> [H];  "                           NEGATED
         [X, X, X, H, X, X, X] -> [H];  "                           NEGATED

                  "TEST VECTORS OF DSACK0 CYCLE COMPLETION SIGNAL
test_vectors
([CLK,  DLY0,DLY6,ENB,  A21,A6,AS,  CRTCS,DUDCS,DUDTACK,  ROMCS,  MUXEN,A20]
                                                          -> [DSACK0])
"                        C D D    R    M         D
"    D D                 R U T    O    U         S
" C  L L E    A A A       T D A    M    X A       C
" L  Y Y N    2 6 S       C C C    C    E 2       K
" K  0 6 B    1           S S K    S    N 0       0
"=============================================================
 [CK,L,H,H,   L,X,L,   H,H,H,   H,   H,L]  -> [L];  "32 BIT COMPLETION (RAM)"
 [L, L,H,H,   L,X,H,   H,H,H,   H,   H,L]  -> [H];  "NO COMPLETION

 [L, L,H,H,   L,X,L,   H,H,H,   L,   H,L]  -> [L];  "8 BIT COMPLETION (ROM)
 [L, L,L,H,   L,X,H,   H,H,H,   L,   H,L]  -> [H];  "NO COMPLETION
 [L, L,L,H,   L,X,L,   H,H,H,   H,   H,H]  -> [H];  "NO COMPLETION

 [CK,H,L,H,   L,X,L,   H,H,H,   H,   L,H]  -> [L];  "8 BIT COMPLETION (MRRAM)

 [CK,L,H,L,   L,X,L,   H,H,H,   H,   H,L]  -> [L];  "32 BIT COMPLETION (A20=L)"
 [L, L,H,H,   L,X,L,   H,H,H,   H,   H,H]  -> [H];  "NO COMPLETION

 [L, L,H,H,   H,L,L,   L,H,H,   H,   H,H]  -> [L];  "8 BIT COMPLETION (CRT.CON.)
 [CK,H,L,H,   H,L,H,   H,H,H,   H,   H,H]  -> [H];  "NO COMPLETION

 [CK,L,H,L,   H,H,L,   H,L,L,   H,   H,H]  -> [L];  "8 BIT COMPLETION (DUART)
 [L, L,H,H,   H,H,L,   H,L,H,   H,   H,H]  -> [H];  "NO COMPLETION
 [L, L,H,H,   H,H,H,   H,H,L,   H,   H,H]  -> [H];  "NO COMPLETION

 [CK,H,L,H,   L,X,L,   H,H,H,   H,   L,H]  -> [L];  "8 BIT COMPLETION (MUXEN)
 [L, H,L,L,   L,X,H,   H,H,H,   H,   H,H]  -> [H];  "NO COMPLETION
```

```
                    "TEST VECTORS OF DSACK1 CYCLE COMPLETION SIGNAL
test_vectors ([A21,A6,AS,ROMCS,CRTCS,DUDCS,A20] -> [DSACK1])
"                   ========================================
                    [L,  H, L,   H,    H,    H,    L] -> [L]; "32 BIT COMPLETION
                    [L,  H, L,   L,    H,    H,    L] -> [H]; "NO COMPLETION
                    [H,  L, L,   H,    L,    H,    L] -> [H]; "NO COMPLETION
                    [H,  H, L,   H,    H,    L,    L] -> [H]; "NO COMPLETION
                    [L,  H, L,   H,    H,    H,    H] -> [H]; "NO COMPLETION
                    [L,  L, H,   H,    H,    H,    L] -> [H]; "NO COMPLETION



                          "TEST VECTORS OF INIT OUT
test_vectors ([CLK,RESET,A21,A6,AS,DUDCS,INIT] -> [INIT])
"                   ========================================
                    [CK,   H,   H,  H, L,  L,    L ] -> [H];  "NO INITIALIZATION
                    [CK,   L,   X,  X, H,  H,    H ] -> [L];  "INITIAL. IN PROGRESS
                    [CK,   L,   X,  X, H,  H,    L ] -> [L];  "INITIAL. IN PROGRESS
                    [CK,   L,   X,  X, H,  H,    L ] -> [L];  "INITIAL. IN PROGRESS
                    [CK,   H,   X,  X, H,  H,    L ] -> [L];  "INITIAL. IN PROGRESS
                    [CK,   H,   X,  X, H,  H,    L ] -> [L];  "INITIAL. IN PROGRESS
                    [CK,   H,   X,  X, H,  H,    L ] -> [L];  "INITIAL. IN PROGRESS
                    [CK,   L,   X,  X, H,  H,    L ] -> [L];  "INITIAL. IN PROGRESS
                    [CK,   L,   X,  X, H,  H,    L ] -> [L];  "INITIAL. IN PROGRESS
                    [CK,   L,   H,  H, L,  L,    L ] -> [H];  "INITIAL. TERMINATED
                    [CK,   L,   H,  H, L,  L,    H ] -> [H];  "NO INITIALIZATION
                    [CK,   H,   X,  X, H,  H,    H ] -> [H];  "NO INITIALIZATION
                    [CK,   L,   X,  X, H,  H,    H ] -> [L];  "EXTRA INITIALIZATION-
              "ATTEMPT, TO AVOID FROM THIS STATE NEGATE THE RESET SIGNAL AT THE
              "BEGINNING OF INITIALIZATION, AS FOLLOW:"
                    [CK,   L,   X,  X, H,  H,    L ] -> [L];"INITIALIZTN. JUST BEGAN
                    [CK,   H,   X,  X, H,  H,    L ] -> [L];"RESET NEG.,INITL.GOES ON
                    [CK,   H,   H,  H, L,  L,    L ] -> [H];"INITIALIZATION TERMINATED
                    [CK,   H,   X,  X, H,  H,    H ] -> [H];"NO EXTRA INITIALIZATION

                          "TEST VECTORS OF DUDCS OUT
test_vectors ([A21,  A6,  AS] -> [DUDCS])
"                   ==========================
                    [H,    H,   L]  -> [L];     " DUART CHIP SELECT ASSERTED
                    [H,    H,   H]  -> [H];     "                   NEGATED
                    [L,    H,   L]  -> [H];     "                   NEGATED
                    [H,    L,   L]  -> [H];     "                   NEGATED
                    [L,    L,   L]  -> [H];     "                   NEGATED
                    [L,    L,   H]  -> [H];     "                   NEGATED
                    [H,    L,   H]  -> [H];     "                   NEGATED
                    [L,    H,   H]  -> [H];     "                   NEGATED

                          "TEST VECTORS OF DUDWE OUT "
test_vectors ([A21,  A6,  DS,  RW] -> [DUDWE])
"                   ==============================
                    [H,    H,   L,   L]  -> [L];" DUART WRITE ENABLE ASSERTED
                    [H,    H,   H,   L]  -> [H];"                    NEGATED
                    [H,    H,   L,   H]  -> [H];"                    NEGATED
                    [H,    L,   L,   L]  -> [H];"                    NEGATED
                    [L,    H,   L,   L]  -> [H];"                    NEGATED
                    [L,    L,   L,   L]  -> [H];"                    NEGATED
                    [H,    H,   H,   H]  -> [H];"                    NEGATED
```

```
                              "TEST VECTORS OF CRTCS OUT
test_vectors ([A21,   A6,   AS] -> [CRTCS])
"              ============================
               [H,     L,    L]   -> [L];      " CRT CONTROLLER CHIP SELECT ASSERTED
               [H,     L,    H]   -> [H];      "                            NEGATED
               [H,     H,    L]   -> [H];      "                            NEGATED
               [H,     H,    H]   -> [H];      "                            NEGATED
               [L,     H,    H]   -> [H];      "                            NEGATED
               [L,     L,    H]   -> [H];      "                            NEGATED
               [L,     L,    L]   -> [H];      "                            NEGATED


                              "TEST VECTORS OF CRTWE OUT
test_vectors ([A21,   A6,   DS,   RW] -> [CRTWE])
"              ================================
               [H,     L,    L,    L]  ->  [L];" CRT CONTROLLER WRITE ENABLE ASSERTED
               [H,     L,    L,    H]  ->  [H];"                             NEGATED
               [H,     L,    H,    L]  ->  [H];"                             NEGATED
               [H,     L,    H,    L]  ->  [H];"                             NEGATED
               [H,     H,    X,    X]  ->  [H];"                             NEGATED
               [L,     H,    X,    X]  ->  [H];"                             NEGATED
               [L,     L,    X,    X]  ->  [H];"                             NEGATED
               [H,     H,    H,    H]  ->  [H];"                             NEGATED
               [L,     L,    L,    L]  ->  [H];"                             NEGATED

end
```

B. PAL SELIN.DOC FILE


```
ABEL(tm) 3.00a  -  Document Generator              18-Sep-90 11:13 AM
PAL ADDRESS DECODER1 AND CHIP CONTROLLER PROGRAMMING FOR USING IN PAL
22V10 FOR THESIS RESEARCH                                      JAN 17 91
Symbol list for Module PAL_SELIN

A20                             Pin 3  pos, com
A21                             Pin 4  pos, com
A6                              Pin 2  pos, com
AS                              Pin 5  pos, com
CK                              (.C.)
CLK                             Pin 1  pos, com
CRTCS                           Pin 21  pos, com, feed_pin
CRTWE                           Pin 20
DLY0                            Pin 10  pos, com
DLY6                            Pin 11  pos, com
DS                              Pin 6  pos, com
DSACK0                          Pin 15
DSACK1                          Pin 14
DUDCS                           Pin 23  pos, com, feed_pin
DUDTACK                         Pin 8  pos, com
DUDWE                           Pin 22
ENB                             Pin 17  pos, feed_pin
H                               (1)
INHIB                           Pin 18
INIT                            Pin 19  pos, feed_pin
L                               (0)
MUXEN                           Pin 16  pos, com, feed_pin
PAL_SELIN                       Module Name
RESET                           Pin 9  pos, com
ROMCS                           Pin 13  pos, com
RW                              Pin 7  pos, com
S0                              (0)
S1                              (1)
SELIN                           device P22V10
X                               (.X.)
_CRTCS_QN                       Node 34  pos, com
_CRTWE_QN                       Node 33  pos, com
_DSACK0_QN                      Node 28  pos, com
_DSACK1_PR                      Node 26  pos, com
_DSACK1_QN                      Node 27  pos, com
_DSACK1_RE                      Node 25  pos, com
_DUDCS_QN                       Node 36  pos, com
_DUDWE_QN                       Node 35  pos, com
_ENB_QN                         Node 30  pos, com
_INHIB_QN                       Node 31  pos, com
_INIT_QN                        Node 32  pos, com
_MUXEN_QN                       Node 29  pos, com
```

ABEL(tm) 3.00a - Document Generator 18-Sep-90 11:13 AM
PAL ADDRESS DECODER1 AND CHIP CONTROLLER PROGRAMMING FOR USING IN PAL
22V10 FOR THESIS RESEARCH JAN 17 91
Equations for Module PAL_SELIN

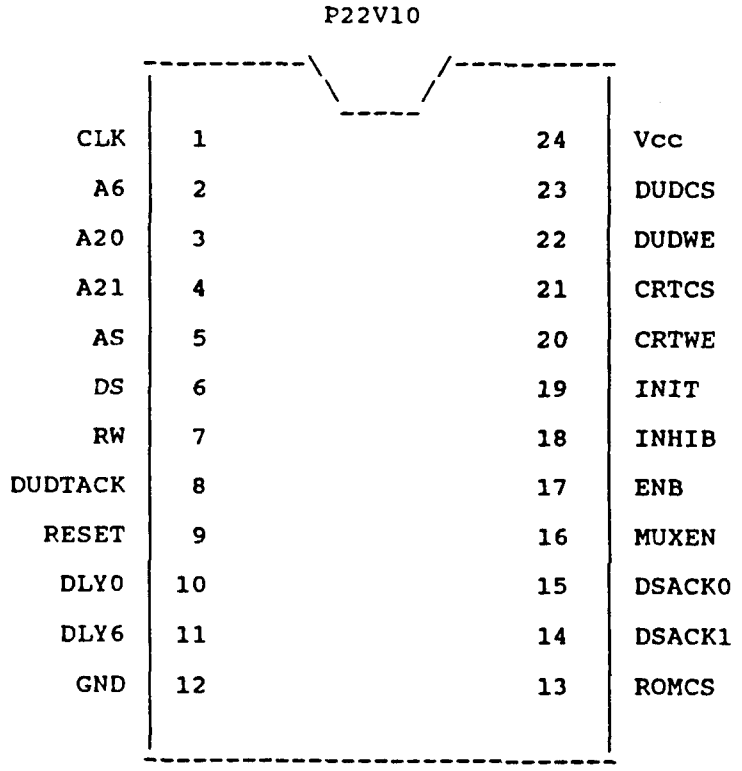
Device SELIN

- Reduced Equations:

```
    ENB := (!DLY0 & ENB # DLY6 & !ENB);

    INHIB = (!AS & DLY0);

    INIT := (!DUDCS # INIT & RESET);

    DSACK0 = !(!MUXEN
              # !DUDCS & !DUDTACK
              # !CRTCS & ENB
              # DLY6 & !ROMCS
              # !A20 & !AS & CRTCS & DUDCS & ROMCS);


    DSACK1 = !(!A20 & !AS & CRTCS & DUDCS & ROMCS);

    MUXEN = (AS # ENB # !A20 # A21);

    DUDCS = (AS # !A6 # !A21);

    DUDWE = !(A21 & A6 & !DS & !RW);

    CRTCS = (AS # A6 # !A21);

    CRTWE = !(A21 & !A6 & !DS & !RW);
```

ABEL(tm) 3.00a - Document Generator 18-Sep-90 11:13 AM
PAL ADDRESS DECODER1 AND CHIP CONTROLLER PROGRAMMING FOR USING IN PAL
22V10 FOR THESIS RESEARCH JAN 17 91
Chip diagram for Module PAL_SELIN

Device SELIN



ABEL(tm) 3.00a - Document Generator 18-Sep-90 11:13 AM
PAL ADDRESS DECODER1 AND CHIP CONTROLLER PROGRAMMING FOR USING IN PAL
22V10 FOR THESIS RESEARCH JAN 17 91
Fuse Map for Module PAL_SELIN

Device SELIN

```
       0          10         20         30         40

   44: ---------- ---------- ---------- ---------- ----
   88: ---------- ------X--- ---------- ---------- ----
  132: -----X---- ---------- ---------- ---------- ----
  176: ---------- ---X------ ---------- ---------- ----

  440: ---------- ---------- ---------- ---------- ----
  484: ----X----- --X------- -X---X---- ---------- ----

  924: ---------- ---------- ---------- ---------- ----
  968: ---------- ------X--- ---------- ---------- ----
 1012: ----X----- ---------- ---------- ---------- ----
 1056: ---------- ---X------ ---------- ---------- ----

 1496: ---------- ---------- ---------- ---------- ----
 1540: -----X---- --X------- -X---X---- ---------- ----

 2156: ---------- ---------- ---------- ---------- ----
 2200: ---X------ ---------- ---------- ---------- ----
 2244: ---------- ---------X ---------- --X------- ----

 2904: ---------- ---------- ---------- ---------- ----
 2948: ---------- -------X-- ---------- ------X--- ----

 3652: ---------- ---------- ---------- ---------- ----
 3696: ---------- ---------- -------X-- -------X-- ----
 3740: ---------- ---------- ------X--- ---------- X---

 4312: ---------- ---------- ---------- ---------- ----
 4356: ---------- ------X--- ---------- ---------- ----
 4400: ---------- ---------- -------X-- ---------- ----
 4444: ---------X ---------- ---------- ---------- ----
 4488: ---------- --X------- ---------- ---------- ----

 4884: ---------- ---------- ---------- ---------- ----
 4928: ---------- ---------- ---------- -X-------- ----
 4972: ---X------ ---------- ---------X ---------- ----
 5016: ---------- -X-------- -------X-- ---------- ----
 5060: ---------- ---------- ---------- ---------- X--X
 5104: --X------X X------X-- ---------- ---------- --X-

 5368: ---------- ---------- ---------- ---------- ----
 5412: --X------X X------X-- ---------- ---------- --X-
       0          10
 5808: --X---X--X ---X--X-X-
```

ABEL(tm) 3.00a - Document Generator 18-Sep-90 11:13 AM
PAL ADDRESS DECODER1 AND CHIP CONTROLLER PROGRAMMING FOR USING IN PAL 22V10 FOR THESIS RESEARCH JAN 17 91
for Module PAL_SELIN

Device SELIN

Device Type: P22V10 Terms Used: 33 out of 132

Pin #	Name	Terms Used	Max	Term Type	Pin Type
1	CLK	--	--	---	Clock
2	A6	--	--	---	Input
3	A20	--	--	---	Input
4	A21	--	--	---	Input
5	AS	--	--	---	Input
6	DS	--	--	---	Input
7	RW	--	--	---	Input
8	DUDTACK	--	--	---	Input
9	RESET	--	--	---	Input
10	DLY0	--	--	---	Input
11	DLY6	--	--	---	Input
12	GND	--	--	---	GND
13	ROMCS	--	--	---	Input
14	DSACK1	1	8	Normal	I/O
15	DSACK0	5	10	Normal	I/O
16	MUXEN	4	12	Normal	I/O
17	ENB	2	14	Normal	I/O
18	INHIB	1	16	Normal	I/O
19	INIT	2	16	Normal	I/O
20	CRTWE	1	14	Normal	I/O
21	CRTCS	3	12	Normal	I/O
22	DUDWE	1	10	Normal	I/O
23	DUDCS	3	8	Normal	I/O
24	Vcc	--	--	---	VCC
25	_DSACK1_RE	0	1	Normal	Output (node)
26	_DSACK1_PR	0	1	Normal	Output (node)
27	_DSACK1_QN	--	--	---	Input (node)
28	_DSACK0_QN	--	--	---	Input (node)
29	_MUXEN_QN	--	--	---	Input (node)
30	_ENB_QN	--	--	---	Input (node)
31	_INHIB_QN	--	--	---	Input (node)
32	_INIT_QN	--	--	---	Input (node)
33	_CRTWE_QN	--	--	---	Input (node)
34	_CRTCS_QN	--	--	---	Input (node)
35	_DUDWE_QN	--	--	---	Input (node)
36	_DUDCS_QN	--	--	---	Input (node)

ABEL(tm) 3.00a - Document Generator 18-Sep-90 11:13 AM
PAL ADDRESS DECODER1 AND CHIP CONTROLLER PROGRAMMING FOR USING IN PAL
22V10 FOR THESIS RESEARCH JAN 17 91
Test Vectors for Module PAL_SELIN

Device SELIN

```
  1 [---- 0--- -0-- ---- ---- ---- ---- ---- ----] ->
          [---- ---- ---- ---- -L-- ---- ---- ---- ----];
  2 [---- 1--- -0-- ---- ---- ---- ---- ---- ----] ->
          [---- ---- ---- ---- -L-- ---- ---- ---- ----];
  3 [---- 1--- -1-- ---- ---- ---- ---- ---- ----] ->
          [---- ---- ---- ---- -L-- ---- ---- ---- ----];
  4 [---- 0--- -1-- ---- ---- ---- ---- ---- ----] ->
          [---- ---- ---- ---- -H-- ---- ---- ---- ----];
  5 [C--- ---- -10- ---- 1--- ---- ---- ---- ----] ->
          [---- ---- ---- ---- L--- ---- ---- ---- ----];
  6 [C--- ---- -00- ---- 0--- ---- ---- ---- ----] ->
          [---- ---- ---- ---- L--- ---- ---- ---- ----];
  7 [C--- ---- -10- ---- 0--- ---- ---- ---- ----] ->
          [---- ---- ---- ---- L--- ---- ---- ---- ----];
  8 [C--- ---- -01- ---- 0--- ---- ---- ---- ----] ->
          [---- ---- ---- ---- H--- ---- ---- ---- ----];
  9 [C--- ---- -00- ---- 1--- ---- ---- ---- ----] ->
          [---- ---- ---- ---- H--- ---- ---- ---- ----];
 10 [C--- ---- -00- ---- 1--- ---- ---- ---- ----] ->
          [---- ---- ---- ---- H--- ---- ---- ---- ----];
 11 [C-10 0--- -10- ---- 1--- ---- ---- ---- ----] ->
          [---- ---- ---- ---L ---- ---- ---- ---- ----];
 12 [C-00 0--- -10- ---- 0--- ---- ---- ---- ----] ->
          [---- ---- ---- ---H ---- ---- ---- ---- ----];
 13 [C-10 1--- -10- ---- 0--- ---- ---- ---- ----] ->
          [---- ---- ---- ---H ---- ---- ---- ---- ----];
 14 [C-00 1--- -10- ---- 0--- ---- ---- ---- ----] ->
          [---- ---- ---- ---H ---- ---- ---- ---- ----];
 15 [C-10 0--- -01- ---- 0--- ---- ---- ---- ----] ->
          [---- ---- ---- ---H ---- ---- ---- ---- ----];
 16 [0-00 0--- -01- ---- 1--- ---- ---- ---- ----] ->
          [---- ---- ---- ---H ---- ---- ---- ---- ----];
 17 [0-10 1--- -01- ---- 1--- ---- ---- ---- ----] ->
          [---- ---- ---- ---H ---- ---- ---- ---- ----];
 18 [0-00 1--- -01- ---- 1--- ---- ---- ---- ----] ->
          [---- ---- ---- ---H ---- ---- ---- ---- ----];
 19 [X-X1 X--- -XX- ---- X--- ---- ---- ---- ----] ->
          [---- ---- ---- ---H ---- ---- ---- ---- ----];
 20 [CX00 0--1 -01- 1--1 1--- 1-1- ---- ---- ----] ->
          [---- ---- ---- --L- ---- ---- ---- ---- ----];
 21 [0X00 1--1 -01- 1--1 1--- 1-1- ---- ---- ----] ->
          [---- ---- ---- --H- ---- ---- ---- ---- ----];
 22 [0X00 0--1 -01- 0--1 1--- 1-1- ---- ---- ----] ->
          [---- ---- ---- --L- ---- ---- ---- ---- ----];
 23 [0X00 1--1 -00- 0--1 1--- 1-1- ---- ---- ----] ->
          [---- ---- ---- --H- ---- ---- ---- ---- ----];
 24 [0X10 0--1 -00- 1--1 1--- 1-1- ---- ---- ----] ->
          [---- ---- ---- --H- ---- ---- ---- ---- ----];
 25 [CX10 0--1 -10- 1--0 1--- 1-1- ---- ---- ----] ->
```

ABEL(tm) 3.00a - Document Generator 18-Sep-90 11:13 AM
PAL ADDRESS DECODER1 AND CHIP CONTROLLER PROGRAMMING FOR USING IN PAL
22V10 FOR THESIS RESEARCH JAN 17 91
Test Vectors for Module PAL_SELIN

Device SELIN

```
                [---- ---- ---- --L- ---- ---- ---- ---- ----];
  26 [CX00 0--1 -01- 1--1 0--- 1-1- ---- ---- ----] ->
                [---- ---- ---- --L- ---- ---- ---- ---- ----];
  27 [0X10 0--1 -01- 1--1 1--- 1-1- ---- ---- ----] ->
                [---- ---- ---- --H- ---- ---- ---- ---- ----];
  28 [0011 0--1 -01- 1--1 1--- 0-1- ---- ---- ----] ->
                [---- ---- ---- --L- ---- ---- ---- ---- ----];
  29 [C011 1--1 -10- 1--1 1--- 1-1- ---- ---- ----] ->
                [---- ---- ---- --H- ---- ---- ---- ---- ----];
  30 [C111 0--0 -01- 1--1 0--- 1-0- ---- ---- ----] ->
                [---- ---- ---- --L- ---- ---- ---- ---- ----];
  31 [0111 0--1 -01- 1--1 1--- 1-0- ---- ---- ----] ->
                [---- -- - ---- --H- ---- ---- ---- ---- ----];
  32 [0111 1--0 -01- 1--1 1--- 1-1- ---- ---- ----] ->
                [---- ---- ---- --H- ---- ---- ---- ---- ----];
  33 [CX10 0--1 -10- 1--0 1--- 1-1- ---- ---- ----] ->
                [---- ---- ---- --L- ---- ---- ---- ---- ----];
  34 [0X10 1--1 -10- 1--1 0--- 1-1- ---- ---- ----] ->
                [---- ---- ---- --H- ---- ---- ---- ---- ----];
  35 [-100 0--- ---- 1--- ---- 1-1- ---- ---- ----] ->
                [---- ---- ---- -L-- ---- ---- ---- ---- ----];
  36 [-100 0--- ---- 0--- ---- 1-1- ---- ---- ----] ->
                [---- ---- ---- -H-- ---- ---- ---- ---- ----];
  37 [-001 0--- ---- 1--- ---- 0-1- ---- ---- ----] ->
                [---- ---- ---- -H-- ---- ---- ---- ---- ----];
  38 [-101 0--- ---- 1--- ---- 1-0- ---- ---- ----] ->
                [---- ---- ---- -H-- ---- ---- ---- ---- ----];
  39 [-110 0--- ---- 1--- ---- 1-1- ---- ---- ----] ->
                [---- ---- ---- -H-- ---- ---- ---- ---- ----];
  40 [-000 1--- ---- 1--- ---- 1-1- ---- ---- ----] ->
                [---- ---- ---- -H-- ---- ---- ---- ---- ----];
  41 [C1-1 0--- 1--- ---- --0- --0- ---- ---- ----] ->
                [---- ---- ---- ---- --H- ---- ---- ---- ----];
  42 [CX-X 1--- 0--- ---- --1- --1- ---- ---- ----] ->
                [---- ---- ---- ---- --L- ---- ---- ---- ----];
  43 [CX-X 1--- 0--- ---- --0- --1- ---- ---- ----] ->
                [---- ---- ---- ---- --L- ---- ---- ---- ----];
  44 [CX-X 1--- 0--- ---- --0- --1- ---- ---- ----] ->
                [---- ---- ---- ---- --L- ---- ---- ---- ----];
  45 [CX-X 1--- 1--- ---- --0- --1- ---- ---- ----] ->
                [---- ---- ---- ---- --L- ---- ---- ---- ----];
  46 [CX-X 1--- 1--- ---- --0- --1- ---- ---- ----] ->
                [---- ---- ---- ---- --L- ---- ---- ---- ----];
  47 [CX-X 1--- 1--- ---- --0- --1- ---- ---- ----] ->
                [---- ---- ---- ---- --L- ---- ---- ---- ----];
  48 [CX-X 1--- 0--- ---- --0- --1- ---- ---- ----] ->
                [---- ---- ---- ---- --L- ---- ---- ---- ----];
  49 [CX-X 1--- 0--- ---- --0- --1- ---- ---- ----] ->
                [---- ---- ---- ---- --L- ---- ---- ---- ----];
  50 [C1-1 0--- 0--- ---- --0- --0- ---- ---- ----] ->
```

ABEL(tm) 3.00a - Document Generator 18-Sep-90 11:13 AM
PAL ADDRESS DECODER1 AND CHIP CONTROLLER PROGRAMMING FOR USING IN PAL
22V10 FOR THESIS RESEARCH JAN 17 91
Test Vectors for Module PAL_SELIN

Device SELIN

```
                [---- ---- ---- ---- --H- ---- ---- ---- ----];
 51 [C1-1 0--- 0--- ---- --1- --0- ---- ---- ----] ->
                [---- ---- ---- ---- --H- ---- ---- ---- ----];
 52 [CX-X 1--- 1--- ---- --1- --1- ---- ---- ----] ->
                [---- ---- ---- ---- --H- ---- ---- ---- ----];
 53 [CX-X 1--- 0--- ---- --1- --1- ---- ---- ----] ->
                [---- ---- ---- ---- --L- ---- ---- ---- ----];
 54 [CX-X 1--- 0--- ---- --0- --1- ---- ---- ----] ->
                [---- ---- ---- ---- --L- ---- ---- ---- ----];
 55 [CX-X 1--- 1--- ---- --0- --1- ---- ---- ----] ->
                [---- ---- ---- ---- --L- ---- ---- ---- ----];
 56 [C1-1 0--- 1--- ---- --0- --0- ---- ---- ----] ->
                [---- ---- ---- ---- --H- ---- ---- ---- ----];
 57 [CX-X 1--- 1--- ---- --1- --1- ---- ---- ----] ->
                [---- ---- ---- ---- --H- ---- ---- ---- ----];
 58 [-1-1 0--- ---- ---- ---- ---- ---- ---- ----] ->
                [---- ---- ---- ---- ---- ---- --L- ---- ----];
 59 [-1-1 1--- ---- ---- ---- ---- ---- ---- ----] ->
                [---- ---- ---- ---- ---- ---- --H- ---- ----];
 60 [-1-0 0--- ---- ---- ---- ---- ---- ---- ----] ->
                [---- ---- ---- ---- ---- ---- --H- ---- ----];
 61 [-0-1 0--- ---- ---- ---- ---- ---- ---- ----] ->
                [---- ---- ---- ---- ---- ---- --H- ---- ----];
 62 [-0-0 0--- ---- ---- ---- ---- ---- ---- ----] ->
                [---- ---- ---- ---- ---- ---- --H- ---- ----];
 63 [-0-0 1--- ---- ---- ---- ---- ---- ---- ----] ->
                [---- ---- ---- ---- ---- ---- --H- ---- ----];
 64 [-0-1 1--- ---- ---- ---- ---- ---- ---- ----] ->
                [---- ---- ---- ---- ---- ---- --H- ---- ----];
 65 [-1-0 1--- ---- ---- ---- ---- ---- ---- ----] ->
                [---- ---- ---- ---- ---- ---- --H- ---- ----];
 66 [-1-1 -00- ---- ---- ---- ---- ---- ---- ----] ->
                [---- ---- ---- ---- ---- ---- -L-- ---- ----];
 67 [-1-1 -10- ---- ---- ---- ---- ---- ---- ----] ->
                [---- ---- ---- ---- ---- ---- -H-- ---- ----];
 68 [-1-1 -01- ---- ---- ---- ---- ---- ---- ----] ->
                [---- ---- ---- ---- ---- ---- -H-- ---- ----];
 69 [-0-1 -00- ---- ---- ---- ---- ---- ---- ----] ->
                [---- ---- ---- ---- ---- ---- -H-- ---- ----];
 70 [-1-0 -00- ---- ---- ---- ---- ---- ---- ----] ->
                [---- ---- ---- ---- ---- ---- -H-- ---- ----];
 71 [-0-0 -00- ---- ---- ---- ---- ---- ---- ----] ->
                [---- ---- ---- ---- ---- ---- -H-- ---- ----];
 72 [-1-1 -11- ---- ---- ---- ---- ---- ---- ----] ->
                [---- ---- ---- ---- ---- ---- -H-- ---- ----];
 73 [-0-1 0--- ---- ---- ---- ---- ---- ---- ----] ->
                [---- ---- ---- ---- ---- ---- L--- ---- ----];
 74 [-0-1 1--- ---- ---- ---- ---- ---- ---- ----] ->
                [---- ---- ---- ---- ---- ---- H--- ---- ----];
 75 [-1-1 0--- ---- ---- ---- ---- ---- ---- ----] ->
```

ABEL(tm) 3.00a - Document Generator 18-Sep-90 11:13 AM
PAL ADDRESS DECODER1 AND CHIP CONTROLLER PROGRAMMING FOR USING IN PAL
22V10 FOR THESIS RESEARCH JAN 17 91
Test Vectors for Module PAL_SELIN

Device SELIN

```
            [---- ---- ---- ---- ---- H--- ---- ---- ----];
 76 [-1-1 1--- ---- ---- ---- ---- ---- ---- ----] ->
            [---- ---- ---- ---- ---- H--- ---- ---- ----];
 77 [-1-0 1--- ---- ---- ---- ---- ---- ---- ----] ->
            [---- ---- ---- ---- ---- H--- ---- ---- ----];
 78 [-0-0 1--- ---- ---- ---- ---- ---- ---- ----] ->
            [---- ---- ---- ---- ---- H--- ---- ---- ----];
 79 [-0-0 0--- ---- ---- ---- ---- ---- ---- ----] ->
            [---- ---- ---- ---- ---- H--- ---- ---- ----];
 80 [-0-1 -00- ---- ---- ---- ---- ---- ---- ----] ->
            [---- ---- ---- ---- ---L ---- ---- ---- ----];
 81 [-0-1 -01- ---- ---- ---- ---- ---- ---- ----] ->
            [---- ---- ---- ---- ---H ---- ---- ---- ----];
 82 [-0-1 -10- ---- ---- ---- ---- ---- ---- ----] ->
            [---- ---- ---- ---- ---H ---- ---- ---- ----];
 83 [-0-1 -10- ---- ---- ---- ---- ---- ---- ----] ->
            [---- ---- ---- ---- ---H ---- ---- ---- ----];
 84 [-1-1 -XX- ---- ---- ---- ---- ---- ---- ----] ->
            [---- ---- ---- ---- ---H ---- ---- ---- ----];
 85 [-1-0 -XX- ---- ---- ---- ---- ---- ---- ----] ->
            [---- ---- ---- ---- ---H ---- ---- ---- ----];
 86 [-0-0 -XX- ---- ---- ---- ---- ---- ---- ----] ->
            [---- ---- ---- ---- ---H ---- ---- ---- ----];
 87 [-1-1 -11- ---- ---- ---- ---- ---- ---- ----] ->
            [---- ---- ---- ---- ---H ---- ---- ---- ----];
 88 [-0-0 -00- ---- ---- ---- ---- ---- ---- ----] ->
            [---- ---- ---- ---- ---H ---- ---- ---- ----];
```

end of module PAL_SELIN

APPENDIX E. PAL CAGRI PROGRAM FILES

A. PAL CAGRI.ABL FILE

```
module PAL_CAGRI;
flag '-r3','-f'
title 'PAL ADDRESS DECODER2 AND SYNCHRONIZATION DELAY REGISTER FOR USING IN
PAL 22V10 FOR THESIS RESEARCH                                    JAN 17 91';

    CAGRI  DEVICE  'P22V10';

    CHRCLK,A0,A1,A18,A19,A20      PIN 1,2,3,4,5,6;
    S0,S1,RW,DS,INIT              PIN 7,8,9,10,11;
    EXTSYNC,CRTSYNC,CHRINHIB      PIN 13,14,15;
    MRRAMWE,ROMCS,RAMOE,RAMCS     PIN 16,17,18,19;
    RAMWE4,RAMWE3,RAMWE2,RAMWE1   PIN 20,21,22,23;
    CHRINHIB                      IsType 'reg_D,pos';
    RST                           NODE 25;
    CK,H,L,X = .C.,1,0,.X.;

equations
    CHRINHIB := CRTSYNC;              "HALT THE 6845 CRT HOR. SYNC. SIGNAL
    RST      = CHRINHIB & !EXTSYNC; "UNTIL THE EXTERIOR HOR. SYNC. IS 0

    !MRRAMWE  = ( A20 & !A19 & !A18 & !DS  & !RW);
    !RAMCS    = (!A20 & !A19 & INIT & !DS)# (!INIT & !RW & !DS);
    !RAMOE    = (!A20 & !A19 & INIT &  RW);
    !RAMWE1   = (!A20 & !A19 & !A1  & !A0 & !DS & !RW);

    !RAMWE2   = (!A20 & !A19 & !A1  & !S0 & !DS & !RW)
              # (!A20 & !A19 & !A1  &  A0 & !DS & !RW)
              # (!A20 & !A19 & !A1  &  S1 & !DS & !RW);

    !RAMWE3   = (!A20 & !A19 &  A1  & !A0 & !DS & !RW)
              # (!A20 & !A19 & !A1  & !S1 & !S0 & !DS & !RW)
              # (!A20 & !A19 & !A1  &  S1 &  S0 & !DS & !RW)
              # (!A20 & !A19 & !A1  &  A0 & !S0 & !DS & !RW);

    !RAMWE4   = (!A20 & !A19 &  A0  &  S1 &  S0 & !DS & !RW)
              # (!A20 & !A19 & !S1  & !S0 & !DS & !RW)
              # (!A20 & !A19 &  A1  &  A0 & !DS & !RW)
              # (!A20 & !A19 &  A1  &  S1 & !DS & !RW);

    !ROMCS    = (!A20 & !A19 & !INIT & RW & !DS)
              # (!A20 &  A19 & !A18 &  INIT & RW & !DS);
```

```
"     TEST VECTORS FOR CHRINHIB SYNCHRONIZATION REGISTER
test_vectors  ([CHRCLK,CRTSYNC,EXTSYNC,CHRINHIB] -> [CHRINHIB])
      [CK, L, H, H] -> [L];   "CHR. CLK. COUNTER IS IN PROGRESS
      [CK, L, L, L] -> [L];   "CHR. CLK. COUNTER IS IN PROGRESS
      [CK, L, H, L] -> [L];   "CHR. CLK. COUNTER IS IN PROGRESS
      [CK, H, L, L] -> [L];   "CHRINHIB FIRST BECOMES H, BUT, THEN
                              "SINCE EXTSYNC=L, IT IS RESETTED TO L
      [CK, L, L, H] -> [L];   "CHR. CLK. COUNTER IS IN PROGRESS
      [CK, H, H, L] -> [H];   "INHIBIT THE 6845 CHR. CLK. COUNTER
      [CK, H, H, H] -> [H];   "INHIBIT THE 6845 CHR. CLK. COUNTER
      [CK, H, L, H] -> [L];   "RESET CONDITION IS TRUE, INHIBIT IS OFF


"    TEST VECTORS FOR MRRAM
test_vectors ([A20,A19,A18,INIT,RW,A1,A0,S1,S0,DS] ->
         [MRRAMWE,RAMCS,RAMOE,RAMWE1,RAMWE2,RAMWE3,RAMWE4,ROMCS])
"    A  A  A  I  R  A  A  S  S  D        M  R  R  R  R  R  R  R
"    2  1  1  N  W  1  0  1  0  S        R  A  A  A  A  A  A  O
"    0  9  8  I  |  |  |  |  |  |        A  M  M  M  M  M  M  M
"    |  |  |  T  |  |  |  |  |  |        M  C  O  W  W  W  W  C
"    |  |  |  |  |  |  |  |  |  |        W  S  E  E  E  E  E  S
"    |  |  |  |  |  |  |  |  |  |        E  |  |  1  2  3  4  |
"   ==========================================================
    [H, L, L, X, L, X, X, X, X, L]  -> [ L, X, X, X, X, X, X, X];
    [H, L, L, X, L, X, X, X, X, H]  -> [ H, X, X, X, X, X, X, X];
    [H, L, L, X, H, X, X, X, X, L]  -> [ H, X, X, X, X, X, X, X];
    [H, L, H, X, L, X, X, X, X, L]  -> [ H, X, X, X, X, X, X, X];
    [H, H, H, X, L, X, X, X, X, L]  -> [ H, X, X, X, X, X, X, X];
    [L, H, H, X, L, X, X, X, X, L]  -> [ H, X, X, X, X, X, X, X];


"    TEST VECTORS FOR RAMCS
"    A  A  A  I  R  A  A  S  S  D        M  R  R  R  R  R  R  R
"    2  1  1  N  W  1  0  1  0  S        R  A  A  A  A  A  A  O
"    0  9  8  I  |  |  |  |  |  |        A  M  M  M  M  M  M  M
"    |  |  |  T  |  |  |  |  |  |        M  C  O  W  W  W  W  C
"    |  |  |  |  |  |  |  |  |  |        W  S  E  E  E  E  E  S
"    |  |  |  |  |  |  |  |  |  |        E  |  |  1  2  3  4  |
"   ==========================================================
    [X, X, X, L, L, X, X, X, X, L]  -> [ X, L, X, X, X, X, X, X];
    [X, X, X, L, H, X, X, X, X, L]  -> [ X, H, X, X, X, X, X, X];
    [L, L, X, H, X, X, X, X, X, L]  -> [ X, L, X, X, X, X, X, X];
    [L, L, X, H, X, X, X, X, X, H]  -> [ X, H, X, X, X, X, X, X];
```

```
"     TEST VECTORS FOR RAMOE
"     A  A  A  I  R  A  A  S  S  D          M  R  R  R  R  R  R  R
"     2  1  1  N  W  1  0  1  0  S          R  A  A  A  A  A  A  O
"     0  9  8  I  |  |  |  |  |  |          A  M  M  M  M  M  M  M
"     |  |  |  T  |  |  |  |  |  |          M  C  O  W  W  W  W  C
"     |  |  |  |  |  |  |  |  |  |          W  S  E  E  E  E  E  S
"     |  |  |  |  |  |  |  |  |  |          E  |  |  1  2  3  4  |
"    ================================================================
     [L, L, X, H, H, X, X, X, X, X]  -> [ X, X, L, X, X, X, X, X];
     [L, L, X, L, H, X, X, X, X, X]  -> [ X, X, H, X, X, X, X, X];
     [L, L, X, H, L, X, X, X, X, X]  -> [ X, X, H, X, X, X, X, X];
     [L, H, X, H, H, X, X, X, X, X]  -> [ X, X, H, X, X, X, X, X];

"     TEST VECTORS FOR RAMWE1
"     A  A  A  I  R  A  A  S  S  D          M  R  R  R  R  R  R  R
"     2  1  1  N  W  1  0  1  0  S          R  A  A  A  A  A  A  O
"     0  9  8  I  |  |  |  |  |  |          A  M  M  M  M  M  M  M
"     |  |  |  T  |  |  |  |  |  |          M  C  O  W  W  W  W  C
"     |  |  |  |  |  |  |  |  |  |          W  S  E  E  E  E  E  S
"     |  |  |  |  |  |  |  |  |  |          E  |  |  1  2  3  4  |
"    ================================================================
     [L, L, X, X, L, L, L, X, X, L]  -> [ X, X, X, L, X, X, X, X];
     [L, L, X, X, L, H, L, X, X, L]  -> [ X, X, X, H, X, X, X, X];
     [L, L, X, X, L, L, H, X, X, L]  -> [ X, X, X, H, X, X, X, X];
     [L, L, X, H, H, L, L, X, X, L]  -> [ X, X, X, H, X, X, X, X];

"     TEST VECTORS FOR RAMWE2
"     A  A  A  I  R  A  A  S  S  D          M  R  R  R  R  R  R  R
"     2  1  1  N  W  1  0  1  0  S          R  A  A  A  A  A  A  O
"     0  9  8  I  |  |  |  |  |  |          A  M  M  M  M  M  M  M
"     |  |  |  T  |  |  |  |  |  |          M  C  O  W  W  W  W  C
"     |  |  |  |  |  |  |  |  |  |          W  S  E  E  E  E  E  S
"     |  |  |  |  |  |  |  |  |  |          E  |  |  1  2  3  4  |
"    ================================================================
     [L, L, X, X, L, L, X, X, L, L]  -> [ X, X, X, X, L, X, X, X];
     [L, L, X, X, L, L, H, X, X, L]  -> [ X, X, X, X, L, X, X, X];
     [L, L, X, X, L, L, X, H, X, L]  -> [ X, X, X, X, L, X, X, X];
     [L, H, X, X, L, L, X, X, L, L]  -> [ X, X, X, X, H, X, X, X];

"     TEST VECTORS FOR RAMWE3
"     A  A  A  I  R  A  A  S  S  D          M  R  R  R  R  R  R  R
"     2  1  1  N  W  1  0  1  0  S          R  A  A  A  A  A  A  O
"     0  9  8  I  |  |  |  |  |  |          A  M  M  M  M  M  M  M
"     |  |  |  T  |  |  |  |  |  |          M  C  O  W  W  W  W  C
"     |  |  |  |  |  |  |  |  |  |          W  S  E  E  E  E  E  S
"     |  |  |  |  |  |  |  |  |  |          E  |  |  1  2  3  4  |
"    ================================================================
     [L, L, X, X, L, H, L, X, X, L]  -> [ X, X, X, X, X, L, X, X];
     [L, L, X, X, L, L, X, L, L, L]  -> [ X, X, X, X, X, L, X, X];
     [L, L, X, X, L, L, X, H, H, L]  -> [ X, X, X, X, X, L, X, X];
     [L, L, X, X, L, L, H, X, L, L]  -> [ X, X, X, X, X, L, X, X];
     [H, L, X, X, L, H, X, X, X, L]  -> [ X, X, X, X, X, H, X, X];
```

```
"     TEST VECTORS FOR RAMWE4
"     A  A  A  I  R  A  A  S  S  D        M  R  R  R  R  R  R  R
"     2  1  1  N  W  1  0  1  0  S        R  A  A  A  A  A  A  O
"     0  9  8  I  |  |  |  |  |  |        A  M  M  M  M  M  M  M
"     |  |  |  T  |  |  |  |  |  |        M  C  O  W  W  W  W  C
"     |  |  |  |  |  |  |  |  |  |        W  S  E  E  E  E  E  S
"     |  |  |  |  |  |  |  |  |  |        E  |  |  1  2  3  4  |
"    ================================================================
     [L, L, X, X, L, X, H, H, H, L]  -> [ X, X, X, X, X, X, L, X];
     [L, L, X, X, L, X, X, L, L, L]  -> [ X, X, X, X, X, X, L, X];
     [L, L, X, X, L, H, H, X, X, L]  -> [ X, X, X, X, X, X, L, X];
     [L, L, X, X, L, H, X, H, X, L]  -> [ X, X, X, X, X, X, L, X];
     [H, L, X, X, L, X, H, H, H, L]  -> [ X, X, X, X, X, X, H, X];


"     TEST VECTORS FOR ROMCS

"     A  A  A  I  R  A  A  S  S  D        M  R  R  R  R  R  R  R
"     2  1  1  N  W  1  0  1  0  S        R  A  A  A  A  A  A  O
"     0  9  8  I  |  |  |  |  |  |        A  M  M  M  M  M  M  M
"     |  |  |  T  |  |  |  |  |  |        M  C  O  W  W  W  W  C
"     |  |  |  |  |  |  |  |  |  |        W  S  E  E  E  E  E  S
"     |  |  |  |  |  |  |  |  |  |        E  |  |  1  2  3  4  |
"    ================================================================
     [L, L, X, L, H, X, X, X, X, L]  -> [ X, X, X, X, X, X, X, L];
     [L, L, X, L, L, X, X, X, X, L]  -> [ X, X, X, X, X, X, X, H];
     [L, H, L, H, H, X, X, X, X, L]  -> [ X, X, X, X, X, X, X, L];
     [L, H, L, H, L, X, X, X, X, L]  -> [ X, X, X, X, X, X, X, H];
     [L, H, X, L, H, X, X, X, X, L]  -> [ X, X, X, X, X, X, X, H];
     [H, L, L, H, H, X, X, X, X, L]  -> [ X, X, X, X, X, X, X, H];

end
```

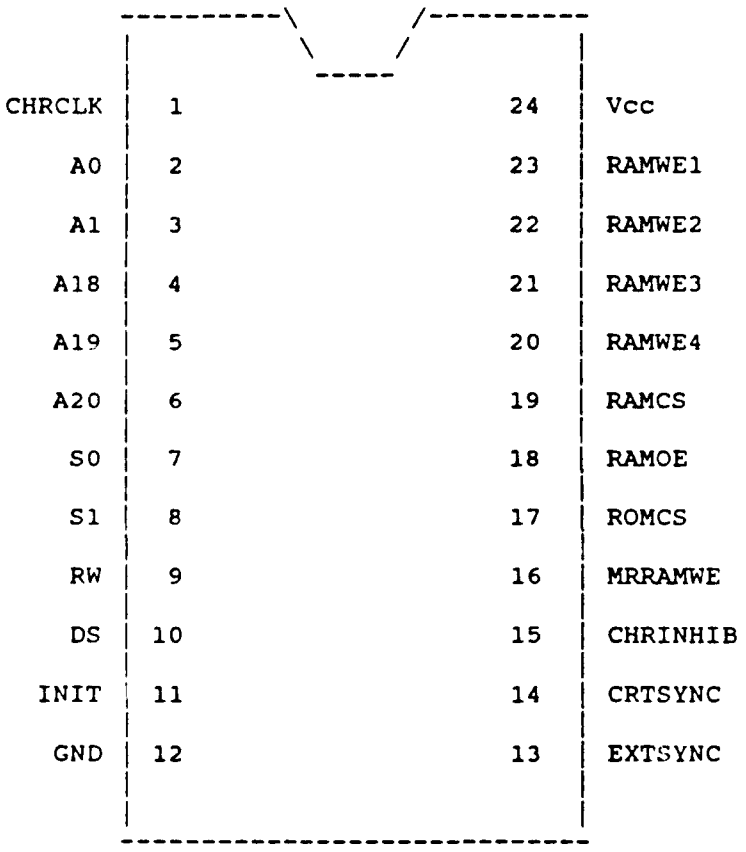
B. PAL CAGRI.DOC FILE

```
                                                                    Page 1
ABEL(tm) Version 2.00b  -  Document Generator          05-Mar-91 03:13 PM
PAL ADDRESS DECODER2 AND SYNCHRONIZATION DELAY REGISTER FOR USING IN
PAL 22V10 FOR THESIS RESEARCH                                   JAN 17 91
Equations for Module PAL_CAGRI

Device CAGRI


     Reduced Equations:

       CHRINHIB := (CRTSYNC);

       RST = (CHRINHIB & !EXTSYNC);

       MRRAMWE = !(!A18 & !A19 & A20 & !DS & !RW);

       RAMCS = !(!DS & !INIT & !RW # !A19 & !A20 & !DS & INIT);

       RAMOE = !(!A19 & !A20 & INIT & RW);

       RAMWE1 = !(!A0 & !A1 & !A19 & !A20 & !DS & !RW);

       RAMWE2 = !(A0 & !A1 & !A19 & !A20 & !DS & !RW
                # !A1 & !A19 & !A20 & !DS & !RW & !S0
                # !A1 & !A19 & !A20 & !DS & !RW & S1);


       RAMWE3 = !(!A0 & A1 & !A19 & !A20 & !DS & !RW
                # A0 & !A1 & !A19 & !A20 & !DS & !RW & !S0
                # !A1 & !A19 & !A20 & !DS & !RW & !S0 & !S1
                # !A1 & !A19 & !A20 & !DS & !RW & S0 & S1);


       RAMWE4 = !(A0 & A1 & !A19 & !A20 & !DS & !RW
                # A1 & !A19 & !A20 & !DS & !RW & S1
                # !A19 & !A20 & !DS & !RW & !S0 & !S1
                # A0 & !A19 & !A20 & !DS & !RW & S0 & S1);


       ROMCS = !(!A19 & !A20 & !DS & !INIT & RW
               # !A18 & A19 & !A20 & !DS & INIT & RW);
```

ABEL(tm) Version 2.00b - Document Generator 05-Mar-91 03:13 PM
PAL ADDRESS DECODER2 AND SYNCHRONIZATION DELAY REGISTER FOR USING IN
PAL 22V10 FOR THESIS RESEARCH JAN 17 91
Chip diagram for Module PAL_CAGRI

Device CAGRI

P22V10



ABEL(tm) Version 2.00b - Document Generator 05-Mar-91 03:13 PM
PAL ADDRESS DECODER2 AND SYNCHRONIZATION DELAY REGISTER FOR USING IN
PAL 22V10 FOR THESIS RESEARCH JAN 17 91
Fuse Map for Module PAL_CAGRI

Device CAGRI

```
      0          10         20         30         40
    0: ---------- ---------- ---------- -----X---- ---X
   44: ---------- ---------- ---------- ---------- ----
   88: -----X---X -------X-- -X-------- ---X---X-- ----
  132: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
  176: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
  220: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
  264: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
  308: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
  352: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
  396: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
  440: ---------- ---------- ---------- ---------- ----
  484: ----X----X -------X-- -X-------- ---X---X-- ----
  528: ---------X -------X-- -X---X---- ---X---X-- ----
  572: ---------X -------X-- -X------X- ---X---X-- ----
  616: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
  660: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
  704: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
  748: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
  792: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
  836: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
  880: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
  924: ---------- ---------- ---------- ---------- ----
  968: -----X--X- -------X-- -X-------- ---X---X-- ----
 1012: ----X----X -------X-- -X---X---- ---X---X-- ----
 1056: ---------X -------X-- -X---X---X ---X---X-- ----
 1100: ---------X -------X-- -X--X---X- ---X---X-- ----
 1144: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 1188: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 1232: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 1276: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 1320: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 1364: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 1408: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 1452: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 1496: ---------- ---------- ---------- ---------- ----
 1540: ----X---X- -------X-- -X-------- ---X---X-- ----
 1584: --------X- -------X-- -X------X- ---X---X-- ----
 1628: ---------- -------X-- -X---X---X ---X---X-- ----
 1672: ----X----- -------X-- -X--X---X- ---X---X-- ----
 1716: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 1760: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 1804: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 1848: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 1892: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
```

```
1936: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
1980: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
2024: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
2068: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
2112: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
```

ABEL(tm) Version 2.00b - Document Generator 05-Mar-91 03:13 PM
PAL ADDRESS DECODER2 AND SYNCHRONIZATION DELAY REGISTER FOR USING IN
PAL 22V10 FOR THESIS RESEARCH JAN 17 91
Fuse Map for Module PAL_CAGRI

Device CAGRI

```
 2156: ---------- ---------- ---------- ---------- ----
 2200: ---------- ---------- ---------- ---X---X-- -X--
 2244: ---------- -------X-- -X-------- -------X-- X---
 2288: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 2332: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 2376: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 2420: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 2464: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 2508: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 2552: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 2596: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 2640: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 2684: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 2728: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 2772: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 2816: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 2860: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 2904: ---------- ---------- ---------- ---------- ----
 2948: ---------- -------X-- -X-------- --X------- X---
 2992: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 3036: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 3080: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 3124: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 3168: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 3212: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 3256: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 3300: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 3344: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 3388: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 3432: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 3476: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 3520: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 3564: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 3608: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 3652: ---------- ---------- ---------- ---------- ----
 3696: ---------- -------X-- -X-------- --X----X-- -X--
 3740: ---------- ---X--X--- -X-------- --X----X-- X---
 3784: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 3828: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 3872: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 3916: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 3960: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 4004: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 4048: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 4092: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
```

```
4136: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
4180: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
4224: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
4268: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
4312: ---------- ---------- ---------- ---------- ----
```

ABEL(tm) Version 2.00b - Document Generator 05-Mar-91 03:13 PM
PAL ADDRESS DECODER2 AND SYNCHRONIZATION DELAY REGISTER FOR USING IN
PAL 22V10 FOR THESIS RESEARCH JAN 17 91
Fuse Map for Module PAL_CAGRI

Device CAGRI

```
 4356: ---------- ---X---X-- X--------- ---X---X-- ----
 4400: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 4444: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 4488: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 4532: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 4576: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 4620: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 4664: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 4708: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 4752: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 4796: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 4840: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 4884: ---------- ---------- ---------- ---------- ----
 4928: ---------- ---------- ---------- --------X- ----
 4972: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 5016: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 5060: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 5104: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 5148: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 5192: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 5236: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 5280: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 5324: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 5368: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 5412: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 5456: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 5500: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 5544: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 5588: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 5632: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 5676: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 5720: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
 5764: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXX
       0          10
 5808: X-X-X-X-X- X-X-X--XX-
```

ABEL(tm) Version 2.00b - Document Generator 05-Mar-91 03:13 PM
PAL ADDRESS DECODER2 AND SYNCHRONIZATION DELAY REGISTER FOR USING IN
PAL 22V10 FOR THESIS RESEARCH JAN 17 91
for Module PAL_CAGRI

Device CAGRI

Device Type: P22V10 Terms Used: 29 out of 132

Pin #	Name	Terms Used	Max	Term Type	Pin Type
1	CHRCLK	--	--	---	Clock
2	A0	--	--	---	Input
3	A1	--	--	---	Input
4	A18	--	--	---	Input
5	A19	--	--	---	Input
6	A20	--	--	---	Input
7	S0	--	--	---	Input
8	S1	--	--	---	Input
9	RW	--	--	---	Input
10	DS	--	--	---	Input
11	INIT	--	--	---	Input
12	GND	--	--	---	GND
13	EXTSYNC	--	--	---	Input
14	CRTSYNC	0	8	Normal	I/O
15	CHRINHIB	1	10	Normal	I/O
16	MRRAMWE	1	12	Normal	I/O
17	ROMCS	2	14	Normal	I/O
18	RAMOE	1	16	Normal	I/O
19	RAMCS	2	16	Normal	I/O
20	RAMWE4	4	14	Normal	I/O
21	RAMWE3	4	12	Normal	I/O
22	RAMWE2	3	10	Normal	I/O
23	RAMWE1	1	8	Normal	I/O
24	Vcc	--	--	---	VCC
25	RST	1	1	Normal	Output (node)
26	_CHRINHIB_PR	0	1	Normal	Output (node)

end of module PAL_CAGRI

APPENDIX F. EPLD SELINJR PROGRAM FILES

A. EPLD SELINJR.ABL FILE

```
module EPLD_SELINJR;
flag '-r3','-f';
title 'EPLD DECADE COUNTER WHICH DRIVES THE ENABLE REGISTER IN PAL SELIN
TO  GENERATE ENABLE SIGNAL.                                    FEB 5 1991';

    SELINJR   DEVICE    'E0310';

        CLK                          PIN 1;
        SERIN,DEL8B,CUREN            PIN 2, 3, 4;
        DISEN,ENBLN,DOTCLK           PIN 5, 6, 7;
        HORSN,VERSN,CLKENB           PIN 8, 9, 11;
        TEXVI,TEXSN,DLY6,DLY0        PIN 12,13,14,15;
        Q3,Q2,Q1,Q0                  PIN 16,17,18,19;
        Q0,Q1,Q2,Q3                  IsType 'pos,reg_D,feed_reg';
        DLY0,DLY6                    IsType 'pos,com';
        H,L,X,CK = 1,0,.X.,.C.;

"STATES OF EPLD DELAY COUNTER
         S0 = ^B0000;
         S1 = ^B0001;
         S2 = ^B0010;
         S3 = ^B0011;
         S4 = ^B0100;
         S5 = ^B0101;
         S6 = ^B0110;
         S7 = ^B0111;
         S8 = ^B1000;
         S9 = ^B1001;

STATE_DIAGRAM [Q3, Q2, Q1, Q0];
         STATE S0 : IF !CLKENB THEN S1 ELSE S0;
         STATE S1 : IF !CLKENB THEN S2 ELSE S1;
         STATE S2 : IF !CLKENB THEN S3 ELSE S2;
         STATE S3 : IF !CLKENB THEN S4 ELSE S3;
         STATE S4 : IF !CLKENB THEN S5 ELSE S4;
         STATE S5 : IF !CLKENB THEN S6 ELSE S5;
         STATE S6 : IF !CLKENB THEN S7 ELSE S6;
         STATE S7 : IF !CLKENB THEN S8 ELSE S7;
         STATE S8 : IF !CLKENB THEN S9 ELSE S8;
         STATE S9 : IF !CLKENB THEN S0 ELSE S9;
```

```
equations
    !TEXVI = ((SERIN $ CUREN $ DEL8B) & ENBLN & DISEN & DOTCLK);
     TEXSN = (HORSN # VERSN);

     DLY0  = (!Q3 & !Q2 & !Q1 & !Q0);
     DLY6  = (!Q3 &  Q2 &  Q1 & !Q0);


"                        TEST VECTORS FOR DELAY COUNTER
test_vectors
          ([CLK, CLKENB, Q3, Q2, Q1, Q0] -> [Q3, Q2, Q1, Q0,   DLY0, DLY6])
"          ================================================================
           [  L,     L,    L,  L,  L,  L] -> [ L,  L,  L,  L,     H,    L ];
           [ CK,     H,    L,  L,  L,  L] -> [ L,  L,  L,  L,     H,    L ];
           [ CK,     L,    L,  L,  L,  L] -> [ L,  L,  L,  H,     L,    L ];
           [ CK,     H,    L,  L,  L,  H] -> [ L,  L,  L,  H,     L,    L ];
           [ CK,     L,    L,  L,  L,  H] -> [ L,  L,  H,  L,     L,    L ];
           [ CK,     H,    L,  L,  H,  L] -> [ L,  L,  H,  L,     L,    L ];
           [ CK,     L,    L,  L,  H,  L] -> [ L,  L,  H,  H,     L,    L ];
           [ CK,     H,    L,  L,  H,  H] -> [ L,  L,  H,  H,     L,    L ];
           [ CK,     L,    L,  L,  H,  H] -> [ L,  H,  L,  L,     L,    L ];
           [ CK,     H,    L,  H,  L,  L] -> [ L,  H,  L,  L,     L,    L ];
           [ CK,     L,    L,  H,  L,  L] -> [ L,  H,  L,  H,     L,    L ];
           [ CK,     H,    L,  H,  L,  H] -> [ L,  H,  L,  H,     L,    L ];
           [ CK,     L,    L,  H,  L,  H] -> [ L,  H,  H,  L,     L,    H ];
           [ CK,     H,    L,  H,  H,  L] -> [ L,  H,  H,  L,     L,    H ];
           [ CK,     L,    L,  H,  H,  L] -> [ L,  H,  H,  H,     L,    L ];
           [ CK,     H,    L,  H,  H,  H] -> [ L,  H,  H,  H,     L,    L ];
           [ CK,     L,    L,  H,  H,  H] -> [ H,  L,  L,  L,     L,    L ];
           [ CK,     H,    H,  L,  L,  L] -> [ H,  L,  L,  L,     L,    L ];
           [ CK,     L,    H,  L,  L,  L] -> [ H,  L,  L,  H,     L,    L ];
           [ CK,     H,    H,  L,  L,  H] -> [ H,  L,  L,  H,     L,    L ];
           [ CK,     L,    H,  L,  L,  H] -> [ L,  L,  L,  L,     H,    L ];
           [ CK,     H,    L,  L,  L,  L] -> [ L,  L,  L,  L,     H,    L ];
           [  L,     L,    L,  L,  L,  L] -> [ L,  L,  L,  L,     H,    L ];
           [ CK,     L,    L,  L,  L,  L] -> [ L,  L,  L,  H,     L,    L ];

"                TEST VECTORS FOR TEXT VIDEO SYNCRONIZATION OUTPUT
test_vectors   ([ HORSN, VERSN ] -> [ TEXSN ])
"              ================================
               [   L,      L    ] -> [   L    ];
               [   L,      H    ] -> [   H    ];
               [   H,      L    ] -> [   H    ];
               [   H,      H    ] -> [   H    ];
```

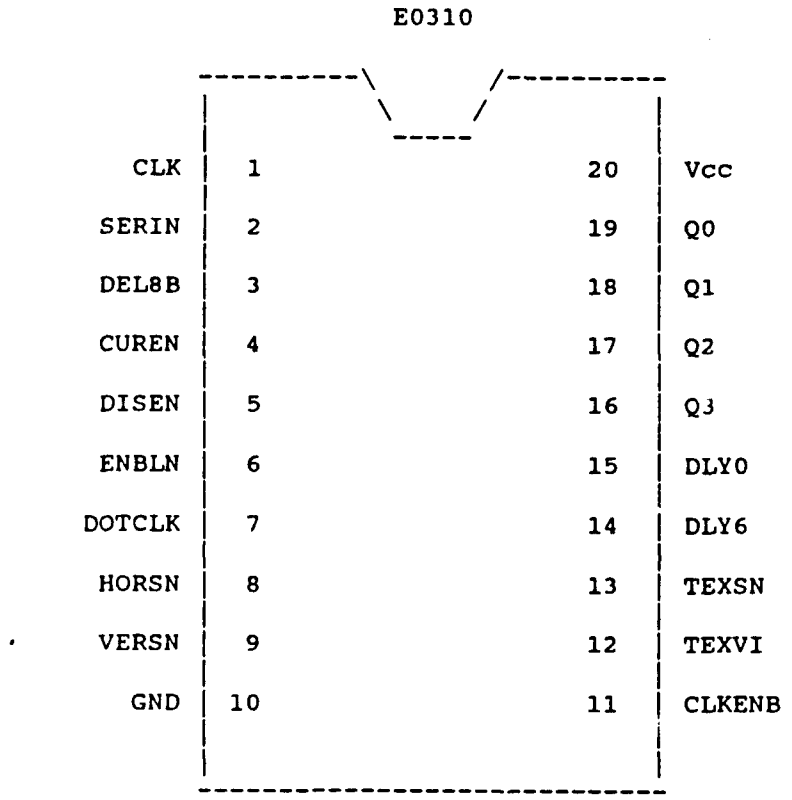
```
"              TEST VECTORS FOR TEXT VIDEO OUTPUT
test_vectors
        ([ SERIN, CUREN, DEL8B, ENBLN, DISEN, DOTCLK]  ->  [ TEXVI ])
"       ==============================================================
        [     X,     X,     X,     X,     X,     L ]  ->  [   H   ];
        [     X,     X,     X,     X,     L,     X ]  ->  [   H   ];
        [     X,     X,     X,     L,     X,     X ]  ->  [   H   ];
        [     X,     X,     X,     L,     L,     L ]  ->  [   H   ];
        [     L,     L,     L,     H,     H,     H ]  ->  [   H   ];
        [     L,     L,     H,     H,     H,     H ]  ->  [   L   ];
        [     L,     H,     L,     H,     H,     H ]  ->  [   L   ];
        [     L,     H,     H,     H,     H,     H ]  ->  [   H   ];
        [     H,     L,     L,     H,     H,     H ]  ->  [   L   ];
        [     H,     L,     H,     H,     H,     H ]  ->  [   H   ];
        [     H,     H,     L,     H,     H,     H ]  ->  [   H   ];
        [     H,     H,     H,     H,     H,     H ]  ->  [   L   ];
end
```

B. EPLD SELINJR.DOC FILE


```
ABEL(tm) Version 2.00b  -  Document Generator            06-Feb-91 07:03 PM
EPLD DECADE COUNTER WHICH DRIVES THE ENABLE REGISTER IN PAL SELIN
TO  GENERATE ENABLE SIGNAL.                                  FEB 5 1991
Equations for Module EPLD_SELINJR

Device SELINJR


     Reduced Equations:

       Q3 := (!CLKENB & Q0 & Q1 & Q2 & !Q3
             # !Q0 & !Q1 & !Q2 & Q3
             # CLKENB & !Q1 & !Q2 & Q3);


       Q2 := (!CLKENB & Q0 & Q1 & !Q2 & !Q3
             # !Q0 & Q2 & !Q3
             # !Q1 & Q2 & !Q3
             # CLKENB & Q2 & !Q3);


       Q1 := (!CLKENB & Q0 & !Q1 & !Q3
             # !Q0 & Q1 & !Q3
             # CLKENB & Q1 & !Q3);


       Q0 := (!CLKENB & !Q0 & !Q1 & !Q2
             # !CLKENB & !Q0 & !Q3
             # CLKENB & Q0 & !Q1 & !Q2
             # CLKENB & Q0 & !Q3);


       TEXVI = !(!CUREN & !DEL8B & DISEN & DOTCLK & ENBLN & SERIN
               # !CUREN & DEL8B & DISEN & DOTCLK & ENBLN & !SERIN
               # CUREN & !DEL8B & DISEN & DOTCLK & ENBLN & !SERIN
               # CUREN & DEL8B & DISEN & DOTCLK & ENBLN & SERIN);


       TEXSN = (HORSN # VERSN);

       DLY0 = (!Q0 & !Q1 & !Q2 & !Q3);

       DLY6 = (!Q0 & Q1 & Q2 & !Q3);
```

ABEL(tm) Version 2.00b - Document Generator 06-Feb-91 07:03 PM
EPLD DECADE COUNTER WHICH DRIVES THE ENABLE REGISTER IN PAL SELIN
TO GENERATE ENABLE SIGNAL. FEB 5 1991
Chip diagram for Module EPLD_SELINJR

Device SELINJR



ABEL(tm) Version 2.00b - Document Generator 06-Feb-91 07:03 PM
EPLD DECADE COUNTER WHICH DRIVES THE ENABLE REGISTER IN PAL SELIN
TO GENERATE ENABLE SIGNAL. FEB 5 1991
Fuse Map for Module EPLD_SELINJR

Device SELINJR

```
       0          10         20         30
   0: --X---X--- X---X----- ---------- ------
  36: --X---X--- --------X- ---------- ------
  72: ---X---X-- X---X----- ---------- ------
 108: ---X---X-- --------X- ---------- ------
 144: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 180: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 216: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 252: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 288: ---------- ---------- ---------- ------
 324: --X----X-- X-------X- ---------- ------
 360: ------X--- -X------X- ---------- ------
 396: ---X------ -X------X- ---------- ------
 432: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 468: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 504: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 540: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 576: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 612: ---------- ---------- ---------- ------
 648: --X----X-- -X--X---X- ---------- ------
 684: ------X--- -----X--X- ---------- ------
 720: ---------- X----X--X- ---------- ------
 756: ---X------ -----X--X- ---------- ------
 792: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 828: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 864: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 900: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 936: ---------- ---------- ---------- ------
 972: --X----X-- -X---X--X- ---------- ------
1008: ------X--- X---X----X ---------- ------
1044: ---X------ X---X----X ---------- ------
1080: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1116: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1152: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1188: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1224: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1260: ---------- ---------- ---------- ------
1296: ------X--- X---X---X- ---------- ------
1332: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1368: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1404: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1440: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1476: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1512: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1548: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
```

```
1584: ---------- ---------- ---------- ------
1620: ------X--- -X---X--X- ---------- ------
1656: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1692: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1728: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
```

ABEL(tm) Version 2.00b - Document Generator 06-Feb-91 07:03 PM
EPLD DECADE COUNTER WHICH DRIVES THE ENABLE REGISTER IN PAL SELIN
TO GENERATE ENABLE SIGNAL. FEB 5 1991
Fuse Map for Module EPLD_SELINJR

Device SELINJR

```
 1764: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 1800: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 1836: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 1872: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 1908: ---------- ---------- ---------- ------
 1944: ---------- ---------- ---------X ------
 1980: ---------- ---------- ---------- ---X--
 2016: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2052: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2088: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2124: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2160: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2196: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2232: ---------- ---------- ---------- ------
 2263: -----X--X- --X----X-- -X---X---- ------
 2304: ----X----X --X----X-- -X---X---- ------
 2340: ----X---X- ---X---X-- -X---X---- ------
 2376: -----X---X ---X---X-- -X---X---- ------
 2412: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2448: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2484: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2520: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2556: ---------- ---------- ---------- ------
 2592: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2628: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
       0
 2664: ---X-X-
 2671: ---X-X-
 2678: ---X-X-
 2685: ---X-X-
 2692: -X-----
 2699: -X-----
 2706: -X-----
 2713: X------
```

ABEL(tm) Version 2.00b - Document Generator 06-Feb-91 07:03 PM
EPLD DECADE COUNTER WHICH DRIVES THE ENABLE REGISTER IN PAL SELIN
TO GENERATE ENABLE SIGNAL. FEB 5 1991
for Module EPLD_SELINJR

Device SELINJR

Device Type: E0310 Terms Used: 30 out of 74

Pin #	Name	Terms Used	Max	Term Type	Pin Type
1	CLK	--	--	---	Clock
2	SERIN	--	--	---	Input
3	DEL8B	--	--	---	Input
4	CUREN	--	--	---	Input
5	DISEN	--	--	---	Input
6	ENBLN	--	--	---	Input
7	DOTCLK	--	--	---	Input
8	HORSN	--	--	---	Input
9	VERSN	--	--	---	Input
10	GND	--	--	---	GND
11	CLKENB	--	--	---	Input
12	TEXVI	4	8	---	Feedback
13	TEXSN	2	8	---	Feedback
14	DLY6	1	8	---	Feedback
15	DLY0	1	8	---	Feedback
16	Q3	3	8	---	Feedback
17	Q2	4	8	---	Feedback
18	Q1	3	8	---	Feedback
19	Q0	4	8	---	Feedback
20	Vcc	--	--	---	VCC
21	_DLY0_RE	0	1	Normal	Output (node)
22	_DLY0_PR	0	1	Normal	Output (node)

end of module EPLD_SELINJR

APPENDIX G. EPLD NESRIN PROGRAM FILES

A. EPLD NESRIN.ABL FILE

```
module EPLD_NESRIN;
flag '-r3','-f';
title 'EPLD SECOND DELAY COUNTER AND DSACK SIGNAL DECODER
TO  GENERATE ENABLE SIGNAL.                                    MAR 8 1991';

    NESRIN   DEVICE    'E0310';

        CLK                          PIN 1;
        I0,I1,I2,I3                  PIN 2, 3, 4, 5;
        AS,ROMCS,CRTCS,DUDCS         PIN 6, 7, 8, 9;
        DUDTACK,MUXEN                PIN 11,12;
        DLY1,DLY2,DSACK1,DSACK0      PIN 13,14,15,16;
        QC,QB,QA                     PIN 17,18,19;
        QC,QB,QA                     IsType 'pos,reg_D,feed_reg';
        DSACK1,DSACK0                IsType 'neg,com';
        H,L,X,CK = 1,0,.X.,.C.;

"STATES OF EPLD DELAY COUNTER
         S0 = ^B000;
         S1 = ^B001;
         S2 = ^B010;
         S3 = ^B011;
         S4 = ^B100;
         S5 = ^B101;
         S6 = ^B110;
         S7 = ^B111;

STATE_DIAGRAM [QC, QB, QA];
         STATE S0 : GOTO S1;
         STATE S1 : GOTO S2;
         STATE S2 : GOTO S3;
         STATE S3 : GOTO S4;
         STATE S4 : GOTO S5;
         STATE S5 : GOTO S6;
         STATE S6 : GOTO S7;
         STATE S7 : GOTO S7;

equations
     !DSACK0 = (!AS & ROMCS & CRTCS & DUDCS & MUXEN)
               #(!ROMCS & (QC & QB & QA))
               #(!CRTCS & DLY2) # (!DUDCS & !DUDTACK)
               # !MUXEN;
     !DSACK1 = (!AS & ROMCS & CRTCS & DUDCS & MUXEN);

     DLY1  = (!I3 & !I2 & !I1 &  I0);
     DLY2  = (!I3 & !I2 &  I1 & !I0);

     [QC.RE,QB.RE,QA.RE] = AS;
```

```
"                TEST VECTORS FOR SECOND DELAY COUNTER
test_vectors
          ([CLK,  AS, QC, QB, QA] -> [QC, QB, QA ])
"          ================================================
           [  L,   L,    L,   L,   L] -> [ L,   L,   L ];
           [ CK,   L,    L,   L,   L] -> [ L,   L,   H ];
           [ CK,   L,    L,   L,   H] -> [ L,   H,   L ];
           [ CK,   L,    L,   H,   L] -> [ L,   H,   H ];
           [ CK,   L,    L,   H,   H] -> [ H,   L,   L ];
           [ CK,   L,    H,   L,   L] -> [ H,   L,   H ];
           [ CK,   L,    H,   L,   H] -> [ H,   H,   L ];
           [ CK,   L,    H,   H,   L] -> [ H,   H,   H ];
           [ CK,   H,    L,   L,   L] -> [ L,   L,   L ];
           [ CK,   H,    L,   L,   L] -> [ L,   L,   L ];
           [ CK,   L,    L,   L,   L] -> [ L,   L,   H ];
           [ CK,   L,    L,   L,   H] -> [ L,   H,   L ];
           [ CK,   H,    L,   L,   L] -> [ L,   L,   L ];
           [  L,   H,    L,   L,   L] -> [ L,   L,   L ];
           [ CK,   L,    L,   L,   L] -> [ L,   L,   H ];

"                TEST VECTORS FOR DLY1 AND DLY2 DELAY DECODERS
test_vectors
          ([  I3,  I2, I1, I0] -> [DLY1, DLY2 ])
"          ============================================================
           [  L,    L,   L,   L ] -> [ L,     L    ];
           [  L,    L,   L,   H ] -> [ H,     L    ];
           [  L,    L,   H,   L ] -> [ L,     H    ];
           [  L,    L,   H,   H ] -> [ L,     L    ];
           [  L,    H,   L,   L ] -> [ L,     L    ];
           [  L,    H,   L,   H ] -> [ L,     L    ];
           [  L,    H,   H,   L ] -> [ L,     L    ];
           [  L,    H,   H,   H ] -> [ L,     L    ];
           [  H,    L,   L,   L ] -> [ L,     L    ];
           [  H,    L,   L,   H ] -> [ L,     L    ];
           [  H,    L,   H,   L ] -> [ L,     L    ];
           [  H,    L,   H,   H ] -> [ L,     L    ];
           [  H,    H,   L,   L ] -> [ L,     L    ];
           [  H,    H,   L,   H ] -> [ L,     L    ];
           [  H,    H,   H,   L ] -> [ L,     L    ];
           [  H,    H,   H,   H ] -> [ L,     L    ];
```

```
                    "TEST VECTORS OF DSACK0 CYCLE COMPLETION SIGNAL
 test_vectors
 ([CLK,QC,QB,QA,  I3,I2,I1,I0, AS, ROMCS,CRTCS,DUDCS,DUDTACK,MUXEN]
                                                          ->[DSACK0])

"                                R   C   D D  M       D
"                                O   R   U U  U       S
"C                               M   T   D D  X       C
"L   Q   Q   Q    I I I I  A     C   C   C T  E       K
"K   C   B   A    3 2 1 0  S     S   S   S C  N       0
"=============================================================================
[L , L,  L,  L,   X,X,X,X, L, H,  H,  H,H, H]  -> [L];"32 BIT COMPLETION (RAM)
[CK, L,  L,  L,   X,X,X,X, H, H,  H,  H,H, H]  -> [H];"NO COMPLETION
[L , H,  H,  H,   H,H,H,H, L, L,  H,  H,H, H]  -> [L];"8 BIT COMPLETION (ROM)
[L , L,  L,  L,   L,L,L,L, H, H,  H,  H,H, H]  -> [H];"NO COMPLETION
[L , H,  L,  L,   L,L,H,H, L, H,  H,  H,H, L]  -> [L];"8 BIT COMPLETION (MRRAM)
[L , L,  L,  L,   H,H,H,H, H, H,  L,  H,H, H]  -> [H];"NO COMPLETION
[L , X,  X,  X,   L,L,H,L, L, H,  L,  H,H, H]  -> [L];"8 BIT COMPLETION (CRT.CN)
[L , L,  L,  L,   H,H,H,H, H, H,  H,  H,H, H]  -> [H];"NO COMPLETION
[L , H,  H,  H,   H,H,H,H, L, H,  H,  L,L, H]  -> [L];"8 BIT COMPLETION (DUART)
[L , L,  L,  L,   H,H,H,H, H, H,  H,  H,H, H]  -> [H];"NO COMPLETION

                  "TEST VECTORS OF DSACK1 CYCLE COMPLETION SIGNAL
test_vectors ([AS,ROMCS,CRTCS,DUDCS,MUXEN]  -> [DSACK1])
"             ================================================
              [ L,   H,    H,    H,    H ] -> [L]; "32 BIT COMPLETION
              [ L,   L,    H,    H,    H ] -> [H]; "NO COMPLETION
              [ L,   H,    L,    H,    H ] -> [H]; "NO COMPLETION
              [ L,   H,    H,    L,    H ] -> [H]; "NO COMPLETION
              [ L,   H,    H,    H,    L ] -> [H]; "NO COMPLETION
              [ H,   X,    X,    X,    X ] -> [H]; "NO COMPLETION

end
```

B. EPLD NESRIN.DOC FILE

ABEL(tm) Version 2.00b - Document Generator 08-Mar-91 10:07 PM
EPLD SECOND DELAY COUNTER AND DSACK SIGNAL DECODER
TO GENERATE ENABLE SIGNAL. MAR 8 1991
Equations for Module EPLD_NESRIN

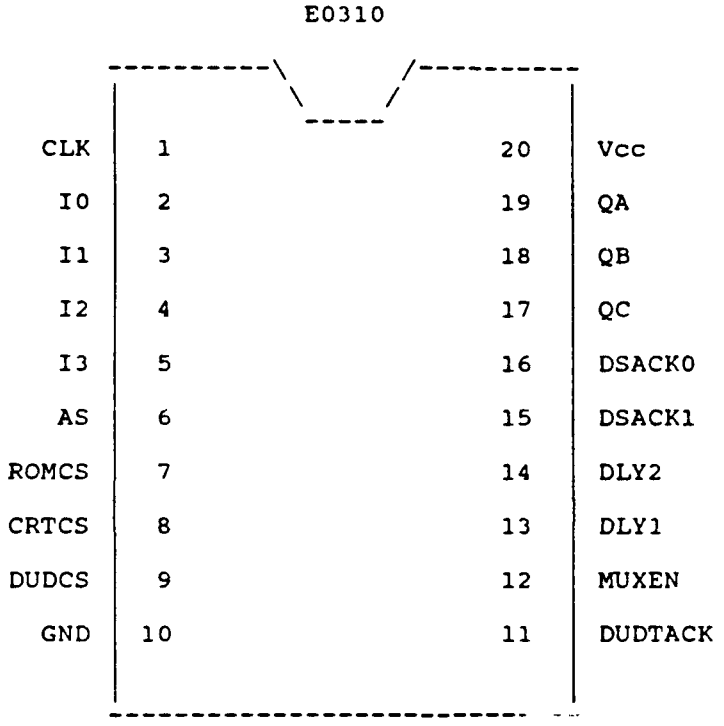
Device NESRIN

Reduced Equations:

```
QC := (QA & QB # QC);

QB := (!QA & QB # QA & !QB # QB & QC);

QA := (!QA # QB & QC);

DSACK0 = !(!MUXEN
          # !CRTCS & DLY2
          # !DUDCS & !DUDTACK
          # QA & QB & QC & !ROMCS
          # !AS & CRTCS & DUDCS & ROMCS);


DSACK1 = !(!AS & CRTCS & DUDCS & MUXEN & ROMCS);

DLY1 = (I0 & !I1 & !I2 & !I3);

DLY2 = (!I0 & I1 & !I2 & !I3);

_DSACK0_RE = (AS);
```

ABEL(tm) Version 2.00b - Document Generator 08-Mar-91 10:07 PM
EPLD SECOND DELAY COUNTER AND DSACK SIGNAL DECODER
TO GENERATE ENABLE SIGNAL. MAR 8 1991
Chip diagram for Module EPLD_NESRIN

Device NESRIN



ABEL(tm) Version 2.00b - Document Generator 08-Mar-91 10:07 PM
EPLD SECOND DELAY COUNTER AND DSACK SIGNAL DECODER
TO GENERATE ENABLE SIGNAL. MAR 8 1991
Fuse Map for Module EPLD_NESRIN

Device NESRIN

```
       0          10         20         30
   0: ------X--- ---------- ---------- ------
  36: ---------- -X---X---- ---------- ------
  72: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 108: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 144: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 180: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 216: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 252: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 288: ---------- ---------- ---------- ------
 324: ------X--- -X-------- ---------- ------
 360: -------X-- X--------- ---------- ------
 396: ---------- -X---X---- ---------- ------
 432: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 468: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 504: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 540: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 576: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 612: ---------- ---------- ---------- ------
 648: -------X-- -X-------- ---------- ------
 684: ---------- -----X---- ---------- ------
 720: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 756: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 792: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 828: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 864: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 900: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 936: ---------- ---------- ---------- ------
 972: ---------- ---------- ---------- ----X-
1008: ---------- ---------- -------XX- ------
1044: --X------- ---------- ---------- --X---
1080: -------X-- -X---X---- ----X----- ------
1116: ---------- ---------- X----X---X ---X--
1152: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1188: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1224: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1260: ---------- ---------- ---------- ------
1296: ---------- ---------- X----X---X ---X-X
1332: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1368: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1404: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1440: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1476: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1512: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1548: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1584: ---------- ---------- ---------- ------
1620: ----X----X --X---X--- ---------- ------
1656: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1692: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
1728: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
```

ABEL(tm) Version 2.00b - Document Generator 08-Mar-91 10:07 PM
EPLD SECOND DELAY COUNTER AND DSACK SIGNAL DECODER
TO GENERATE ENABLE SIGNAL. MAR 8 1991
Fuse Map for Module EPLD_NESRIN

Device NESRIN

```
 1764: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 1800: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 1836: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 1872: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 1908: ---------- ---------- ---------- ------
 1944: -----X--X- --X---X--- ---------- ------
 1980: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2016: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2052: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2088: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2124: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2160: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2196: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2232: ---------- ---------- ---------- ------
 2268: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2304: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2340: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2376: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2412: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2448: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2484: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2520: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2556: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2592: XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXX
 2628: ---------- ---------- -X-------- ------
       0
 2664: ---X-X-
 2671: ---X-X-
 2678: ---X-X-
 2685: X------
 2692: X------
 2699: -X-----
 2706: -X-----
 2713: -------
```

ABEL(tm) Version 2.00b - Document Generator 08-Mar-91 10:07 PM
EPLD SECOND DELAY COUNTER AND DSACK SIGNAL DECODER
TO GENERATE ENABLE SIGNAL. MAR 8 1991
for Module EPLD_NESRIN

Device NESRIN

Device Type: E0310 Terms Used: 23 out of 74

Pin #	Name	Terms Used	Max	Term Type	Pin Type
1	CLK	--	--	---	Clock
2	I0	--	--	---	Input
3	I1	--	--	---	Input
4	I2	--	--	---	Input
5	I3	--	--	---	Input
6	AS	--	--	---	Input
7	ROMCS	--	--	---	Input
8	CRTCS	--	--	---	Input
9	DUDCS	--	--	---	Input
10	GND	--	--	---	GND
11	DUDTACK	--	--	---	Input
12	MUXEN	0	8	---	Feedback
13	DLY1	1	8	---	Feedback
14	DLY2	1	8	---	Feedback
15	DSACK1	1	8	---	Feedback
16	DSACK0	5	8	---	Feedback
17	QC	2	8	---	Feedback
18	QB	3	8	---	Feedback
19	QA	2	8	---	Feedback
20	Vcc	--	--	---	VCC
21	_DSACK0_RE	1	1	Normal	Output (node)
22	_DSACK0_PR	0	1	Normal	Output (node)

end of module EPLD_NESRIN

LIST OF REFERENCES

1. Motorola, Inc., *MC68020 32-BIT MICROPROCESSOR USER'S MANUAL*, Prentice-Hall, Inc., 1990.

2. Werner Hilf and Anton Nausch, *THE 68000 FAMILY, Volume 2*, Prentice-Hall, Inc., 1990.

3. Yavuz Tugcu, *DESIGN AND IMPLEMENTATION OF AN MC68020-BASED EDUCATIONAL COMPUTER BOARD*, Master's Thesis, Naval Postgraduate School, December 1989.

4. Michael Slater, *MICROPROCESSOR-BASED DESIGN*, Prentice-Hall, Inc., 1989.

5. National Semiconductor Corporation, *INTERFACE, BIPOLAR LSI, BIPOLAR MEMORY, PROGRAMMABLE LOGIC*, 1983, pp. (9-13) - (9-21).

6. Motorola, Inc., *MOTOROLA, THE COMPLETE MICROCOMPUTER DATA LIBRARY*, 1978, pp. (1-159) - (1-177).

7. Motorola Semiconductors, Advance Information, *MC68681 DUAL ASYNCHRONOUS RECEIVER/TRANSMITTER (DUART) Motorola, Inc., September, 1985.*

INITIAL DISTRIBUTION LIST

		No. Copies
1.	*Defense Technical Information Center* *Cameron Station* *Alexandria, VA 22304-6145*	2
2.	*Library, Code 52* *Naval Postgraduate School* *Monterey, CA 93943-5002*	2
3.	*Chairman, Code EC* *Department of Electrical and Computer Engineering* *Naval Postgraduate School* *Monterey, CA 93943-5000*	1
4.	*Professor Chyan Yang* *Department of Electrical and Computer Engineering, Code EC/YA* *Naval Postgraduate School* *Monterey, CA 93943-5004*	1
5.	*Professor Frederick W. Terman* *Department of Electrical and Computer Engineering, Code EC/TZ* *Naval Postgraduate School* *Monterey, CA 93943-5004*	1
6.	*Hava Kuvvetleri Komutanligi* *Egitim Daire Bsk.ligi* *Bakanliklar, Ankara / TURKEY*	1
7.	*Hava Harp Okulu Komutanligi* *Okul Kutuphanesi* *Yesilyurt, Istanbul / TURKEY*	1
8.	*3NCU HIBM Komutanligi* *Fabrika Mudurlugu* *Etimesgut, Ankara / TURKEY*	1
9.	*Orta Dogu Teknik Universitesi* *Universite Kutuphanesi* *Balgat, Ankara / TURKEY*	1
10.	*Bogazici Universitesi* *Universite Kutuphanesi* *Bebek, Istanbul / TURKEY*	1
11.	*Kadri Hekimoglu* *DeGol Caddesi No. 1/5* *Tandogan, Ankara / TURKEY*	1

12. *Yavuz TUGCU* *1*
Asagi Eglence Mercimek Sokak
41/20 Etlik, Ankara / TURKEY

13. *Ugur Ozkan* *1*
Haci Zihni Efendi Sok. No. 8/3
Capa, Istanbul / TURKEY